Joint Monitoring and Routing in Wireless Sensor Networks using Robust Identifying Codes

Moshe Laifenfeld*, Ari Trachtenberg*, Reuven Cohen* and David Starobinski*
*Department of Electrical and Computer Engineering
Boston University, Boston, MA 02215
Email:{ moshel,trachten,cohenr,staro@bu.edu}

***Abstract*—Wireless Sensor Networks (WSNs) provide an important means of monitoring the physical world, but their limitations present challenges to fundamental network services such as routing. In this work we utilize an abstraction of WSNs based on the theory of identifying codes. This abstraction has been useful in recent literature for a number of important monitoring problems, such as localization and contamination detection. In our case, we use it to provide a joint infrastructure for efficient and robust monitoring and routing in WSNs. Specifically, we make use of efficient and distributed algorithm for generating robust identifying codes, an NP-hard problem, with a logarithmic performance guarantee based on a reduction to the set k-multicover problem. We also show how this same identifying-code infrastructure provides a natural labeling that can be used for near-optimal routing with very small routing tables. We provide experimental results for various topologies that illustrate the superior performance of our approximation algorithms over previous identifying code heuristics.**

I. Introduction

Sensor networks provide a new and potentially revolutionary means of reaching and monitoring our physical surroundings. Important applications of these networks include environmental monitoring of the life-cycle of trees and various animal species [1, 2] and the structural monitoring of buildings, bridges, or even nuclear power stations [3, 4].

A. Identifying code abstraction

For several fundamental monitoring problems, such as localization [5, 6] and identification of contamination sources [7, 8], the theory of *identifying codes* [9] has been found to provide an extremely useful abstraction. Within this abstraction, a monitoring area is divided into a finite number of regions and modeled as a graph, wherein each vertex represents a different region as in Figure 1. In this model, two vertices are connected by a link if they are within communication range. An identifying code for the graph then corresponds to a subset of vertices where monitoring sensors (*i.e., codewords* of the code) are located, such that each vertex is within the communication range of a different set of monitors (referred to as an *identifying set*). Thus, a collection of monitors in a network forms an identifying code if any given identifying set uniquely identifies a vertex in the graph.

An important benefit of identifying codes is that they allow monitoring of an area without the need to place or activate a (possibly expensive) monitor in each sub-region. Since the size of an identifying code is typically much smaller than that of the original graph, this construction can result in a substantial savings in the number of monitors. Alternatively, for a fixed number of monitors, systems based on identifying codes can achieve much higher resolution and robustness than proximity-based systems, in which each sensor only monitors its surrounding region.

Identifying codes provide also means for quantifying energy/robustness trade-offs through the concept of *robust identifying codes*, introduced in [5]. An identifying code is r-robust if the addition or deletion of up to r codewords in the identifying set of *any* vertex does not change its uniqueness. Thus, with an r-robust code, a monitoring system can continue to function properly even if up to r monitors per locality experience failure. Of course, the size of an r-robust code increases with r (typically linearly).

Despite the importance of identifying codes for sensor monitoring applications, the problem of constructing efficient codes (in terms of size) is still unsolved. Specifically, the problem of finding a minimum identifying code for an arbitrary graph has been shown to be NP-hard [10, 11]. In Ray et al. [5], a simple algorithm called ID-CODE was proposed to generate irreducible codes in which no codeword can be removed without violating the unique identifiability of some vertex. However, in some graphs, the size of the resulting code using the ID-CODE algorithm can be arbitrarily poor [12].

B. Contributions

We begin with a presentation of a polynomial-time approximation algorithm with provable performance guarantees (initially introduced in [13–15]) for the minimum r-robust identifying code problem. Our algorithm, called $\mathtt{rID-LOCAL}$, generates a robust identifying code whose size is guaranteed to be at most $1 + 2\log(n)$ times larger than the optimum, where n is the number of vertices (a sharper bound is provided in Section II). This approximation is obtained through a reduction of the original problem to a minimum set k-cover problem, for which greedy approximations are well known.

Our approximation utilizes only localized information, thus lending itself to two distributed implementations, which we term $\mathtt{rID-SYNC}$ and $\mathtt{rID-ASYNC}$. The first implementa-

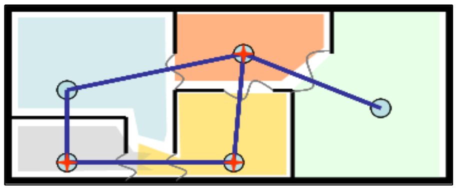
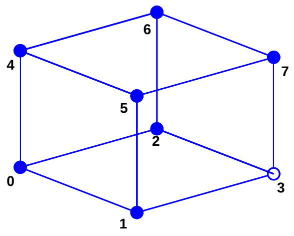


Fig. 1. A floor plan quantized into 5 regions represented by 5 vertices (circles). An edge in the model graph (blue line) represents RF connectivity between a pair of vertices, and the 3 vertices marked by red stars denote an identifying code for the resultant graph.



$$\begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Fig. 2. A 1-robust identifying code for a cube (codewords are solid circles) together with the graph's adjacency matrix; the identifying set of vertex 1 is $\{0, 1, 5\}$.

tion provides a tradeoff between runtime and performance guarantees while using a low communications load and only coarse synchronization; the second implementation requires no synchronization at the expense of more communication. Through simulations and analysis on Erdos-Renyi random graphs and geometric random graphs, we show that these algorithms significantly outperform earlier identifying code algorithms. Our analysis is an extension of [16], which provides and asymptotic bound on the size of an r-robust identifying code in Erdos-Renyi random graphs.

Next, we demonstrate that the identifying code-based monitoring infrastructure can be reused to efficiently implement routing between any two nodes in the network. Specifically, we show how to use routing tables of the same size as the network's identifying code to route packets between any two nodes, within two hops of the shortest path. The significance of this result is two-fold: (i) we can perform near-optimal routing while significantly compressing routing table memory in each sensor; (ii) one algorithm can be run to simultaneously setup both monitoring and routing infrastructures, thus reducing the network setup overhead.

Preliminary version of this work was presented in [15].

C. Outline

In Section II we provide a brief introduction to robust identifying codes followed by a centralized approximation algorithm with proven performance bounds. Thereafter, in Section III we provide and analyze a distributed version of this algorithm. In Section IV we analyze r-robust identifying codes in random graphs, and in Section V we describe a novel technique for reusing identifying codes for routing. Section VI provides some simulation data for the various algorithms considered.

II. Robust identifying codes and the set multicover problem

Given a base set U of m elements and a collection $\mathcal{S}$ of subsets of U, the set cover problem asks to find a minimum sub-collection of S whose elements have U as their union (*i.e.*, they cover U). The set cover problem is one of the oldest and most studied NP-hard problem [17] and it admits a simple greedy approximation: iteratively choose the heretofore unselected set of $\mathcal{S}$ that covers the largest number of uncovered elements in the base set. The classic results of Johnson [18] showed that, for minimum cover s_{min} and greedy cover s_{greedy}, we have that $\frac{s_{greedy}}{s_{min}} = \Theta(\ln m)$. Hardness results [19] suggest that this greedy approach is one of the best polynomial approximations to the problem.

The *minimum set k-multicover* problem is a natural generalization of the set cover problem, in which, given $(U, \mathcal{S})$, we seek the smallest sub-collection of $\mathcal{S}$ that covers every element in *U at least k times* (more formal definitions are in Section II-C). Often this problem is addressed as a special case of the *covering integer problem* [20]. The set k-multicover problem admits a similar greedy heuristic to the set cover problem, with a corresponding performance ratio guarantee [20] of at most $1 + \log\left(\max_{S_i \in \mathcal{S}}(|S_i|)\right)$.

A. Technical definitions

Given an undirected graph $G = (V, E)$, the *ball* $B(v)$ consists of all vertices adjacent to the vertex v, together with v itself. It is possible to generalize this definition (and the corresponding results in the paper) to directed graphs, but this significantly complicates the notation and we omit these extensions for sake of clearer exposition.

A non-empty subset $\mathbb{C} \subseteq V$ is called a *code* and its elements are *codewords*. For a given code $\mathbb{C}$, the *identifying set* $I_{\mathbb{C}}(v)$ of a vertex v is defined to be the codewords neighboring v, *i.e.*, $I_{\mathbb{C}}(v) = B(v) \cap \mathbb{C}$ (if $\mathbb{C}$ is not specified, it is assumed to be the set of all vertices V). A code $\mathbb{C}$ is an *identifying code* if each identifying set of the code is unique, in other words

$$\forall u, v \in V \quad u = v \longleftrightarrow I_{\mathbb{C}}(u) = I_{\mathbb{C}}(v).$$

In our applications, we shall further require that $I_{\mathbb{C}}(v) \neq \emptyset$ for all vertices v, so that an identifying code is also a *vertex cover* or *dominating set*.

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

B. Reduction intuition

Consider a three dimensional cube as in Figure 2 and let $\mathbb{C} = \{0, 1, 2, 4, 5, 6, 7\}$. Clearly, the identifying sets are all unique, and hence the code is an identifying code. A closer look reveals that $\mathbb{C}$ is actually a 1-robust identifying code, so that it remains an identifying code even upon removal or insertion of any vertex into any identifying set.

A graph's adjacency matrix provides a linear algebra view of the identifying code problem. Specifically, we can consider each row and column of the matrix to be a *characteristic vector* of the ball around some vertex in the graph, meaning that their i-th entry of the row j is 1 if and only if the i-th vertex of V is in the ball around node j. Selecting codewords can thus be viewed as selecting columns to form a matrix of size $n \times |C|$. We will refer to this matrix as the *code matrix*. A code is thus *identifying* if the *Hamming distance* between every two rows in the code matrix is at least one (recall that the Hamming distance of two binary vectors is the number of ones in their bitwise XOR). It has been shown in [6] that if the Hamming distance between every two rows in the code matrix is at least $2r+1$ then the set of vertices is r-robust.

We next form the $\frac{n(n-1)}{2} \times n$ *difference matrix* by stacking the bitwise XOR results of every two different rows in the adjacency matrix. The problem of finding a minimum size r-robust identifying code is thus equivalent to finding a minimum number of columns in the difference matrix for which the resulting matrix has minimum Hamming distance $2r+1$ (between any two rows). This equivalent problem is nothing but a set $2r+1$-multicover problem, if one regards the columns of the difference matrix as the characteristic vectors of subsets $\mathcal{S}$ over the base set of all pairs of rows in the adjacency matrix.

In the next subsection we formalize this intuition into a rigorous reduction.

C. Reduction

In this section we formally reduce the problem of finding the smallest sized r-robust identifying code over an arbitrary graph G to a $2r+1$-multicover problem. Formally we connect the following problems:

a) SET MULTI-COVER (SC_k):

INSTANCE:	Subsets S of U, an integer $k \geq 1$.
SOLUTION:	$S' \subseteq S$ such that for every element $u \in U$, $\lvert\{s \in S' : u \in s\}\rvert \geq k$.
MEASURE:	The size of the multicover: $\lvert S'\rvert$.

b) Robust ID-CODE (rID):

INSTANCE:	Graph $G=(V,E)$, and integer $r \geq 0$.
SOLUTION:	An r-robust identifying code $C \subseteq V$.
MEASURE:	The size $\lvert C\rvert$.

Theorem 1 *Given a graph G of n vertices, finding an r-robust identifying code requires no more computations than a $(2r+1)$-multicover solution over a base set of $\frac{n(n-1)}{2}$ elements together with $O(n^3)$ operations of length n binary vectors.*

To prove the theorem we start with a few definitions.

Definition 2 *The* difference set $D_{\mathbb{C}}(u,v)$ *is defined to be the symmetric difference between the identifying sets of vertices $u,v \in V$:*

$$D_{\mathbb{C}}(u,v) \quad \doteq \quad I_{\mathbb{C}}(u) \oplus I_{\mathbb{C}}(v),$$

For simplicity of notation, we shall omit the subscript when looking at identifying codes consisting of all graph vertices, i.e., $D(u,z) = D_V(u,z)$.

Definition 3 *Let* $\mathrm{U} = \{(u,z) | u \neq z, u,z \in V\}$. *Then the* distinguishing set δ_c *is the set of vertex pairs in* U *for which* c *is a member of their difference set:*

$$\delta_c = \{(u,z) \in \mathrm{U} \,|\, c \in D_{\mathbb{C}}(u,z)\}.$$

It has been shown in [6] that a code is r-robust if and only if the size of the smallest difference set is at least $2r+1$. Equivalently, a code is r-robust if and only if its distinguishing sets form a $2r+1$-multicover of all the pairs of vertices in the graph.

Lemma 1 *Given $G=(V,E)$ the following statements are equivalent:*

1) *$\mathbb{C} = \{c_1, ..., c_k\}$ is an r-robust identifying code.*
2) *$|D_{\mathbb{C}}(u,v)| \geq 2r+1$, for all $u \neq v \in V$*
3) *The collection $\{\delta_{c_1}, ..., \delta_{c_k}\}$ forms a $(2r+1)$-multicover of $\mathbf{U} = \{(u,v) \,|\, \forall\, u \neq v \in V\}$.*

Proof of Theorem 1: Given $\mathrm{MinSetCover} : (k, U, S) \to S'$, an algorithm for solving the minimum set-multicover problem, SC_k, consider the following construction of an r-robust identifying code.

ID : $(G, r) \to \mathbb{C}$

1) Compute the identifying sets over V, $\{I(u) | u \in V\}$.
2) Compute the distinguishing sets $\Delta = \{\delta_u | u \in V\}$.
3) Apply the set-multicover algorithm,
   $\mathbf{C} \leftarrow \mathrm{MinSetCover}(2r+1, \mathbf{U}, \Delta)$
4) Output the set of $u \in V$ that correspond to $\delta_u \in \mathbf{C}$, *i.e.,* $\mathbb{C} \leftarrow \{u \in V | \delta_u \in \mathbf{C}\}$

The resulting code, $\mathbb{C}$, is guaranteed by Lemma 1 to be an r-robust identifying code, and the optimality of the set cover in step 3 guarantees that no smaller identifying code can be found. To complete the proof we observe that computing the identifying sets $I(u)$ naively requires $\theta(n^2)$ additions of binary vectors, and computing Δ requires n operations for each of the $\frac{n(n-1)}{2}$ elements in $|\mathbf{U}|$. ∎

D. Localized robust identifying code and its approximation

The construction used in the proof of Theorem 1 together with the well know greedy approximation for the set-multicover problem [20] can be used to derive an approximation to the r-robust identifying code problem.

An example of $\mathtt{rID-CENTRAL}(0, G)$ given in Algorithm 1, where G is a 10 nodes ring is shown in Figure 3, where the optimal identifying code is achieved; however the outcome may vary depending on the way ties are broken and therefore on the labeling scheme of the nodes. In the example of Figure 3 ties are broken in favor of vertices of lower label.

$\mathtt{rID-CENTRAL}$ requires the knowledge of the entire graph in order to operate. It was observed in [6, 21] that an r-robust identifying code can be built in a localized manner, where

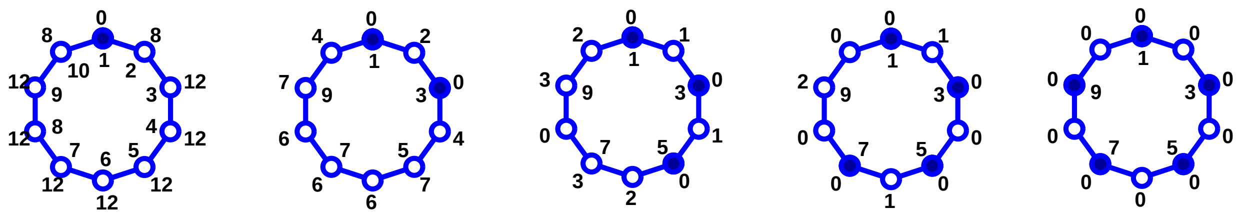


Fig. 3. Demonstration of the $\mathtt{rID-CENTRAL}$ (with $r=0$) for 10 nodes ring, starting on the left. Nodes are labeled 1 to 10 clockwise (the labels appear in the inner perimeter). Solid circles represent codewords, and the distinguishing sets sizes, obtained from the greedy set-multicover (SET-MULTICOVER) iterations in $\mathtt{rID-CENTRAL}$, appear in the outer perimeter. The resultant identifying code (right) can be shown to be optimal.

Algorithm 1 Centralized r-robust code $\mathtt{rID-CENTRAL}(r,G)$

We start with a graph $G=(V,E)$ and a non-negative integer r. The greedy set multicover approximation is denoted SET-MULTICOVER$(k,\mathbf{U},\mathcal{S})$.

```
1) Compute the identifying sets over $V$ $\{I(u)|u\in V\}$
2) Compute the distinguishing sets $\Delta=\{\delta_u|u\in V\}$.
3) Apply $\mathbf{C}\leftarrow$SET-MULTICOVER$(2r+1,\mathbf{U},\Delta)$
4) Output $\mathbb{C}_{central}\leftarrow\{u\in V|\delta_u\in\mathbf{C}\}$
```

each vertex only considers its two-hop neighborhood. The resulting *localized identifying codes* are the subject of this section, and the approximation algorithm we derive is critical to the distributed algorithm of the next section.

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 $\{w\in V|\rho(w,v)\leq l\}$. 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 2 *Given a graph $G=(V,E)$, an $(r+1)$-dominating set $\mathbb{C}$ is also an r-robust identifying code if and only if $|D_{\mathbb{C}}(u,v)|\geq 2r+1$ for all $u,v\in V$ such that $\rho(u,v)\leq 2$.*

Proof: The forward implication is an application of Lemma 1. For the reverse implication we take $\mathbb{C}$ to be an $r+1$ dominating set and assume that $|D_{\mathbb{C}}(u,v)|\geq 2r+1$ for $\rho(u,v)\leq 2$; we will show that this assumption is also valid for $\rho(u,v)>2$. This is because, for $\rho(u,v)>2$, we have that $B(v)\cap B(u)=\emptyset$, meaning that $|D_{\mathbb{C}}(u,v)|=|B(v)\cap\mathbb{C}|+|B(u)\cap\mathbb{C}|$. Since $\mathbb{C}$ is an $r+1$ dominating set, it must be that $|B(y)\cap\mathbb{C}|\geq r+1$ for all vertices y, giving that $|D_{\mathbb{C}}(u,v)|>2r+1$. Applying Lemma 1 we thus see that $\mathbb{C}$ must be r-robust. ∎

The localized robust identifying code approximation Lemma 2 can serve as the basis for a reduction from an identifying code problem to a set cover problem, similarly to Theorem 1. The main difference is that we will restrict 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-robust.

Towards this end we define $\bar{\mathbf{U}}=\{(u,v)\,|\,\rho(u,v)\leq 2\}$, 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 $\bar{\mathbf{U}}$ as follows:

$$\bar{\delta}_v=(\delta_v\cap\bar{\mathbf{U}})\cup\{(u,u)|u\in B(v)\}, \qquad (1)$$

The resulting *localized identifying code approximation* is thus given by Algorithm 2 and can be shown in a similar manner to provide an r-robust identifying code for any graph that admits one.

Algorithm 2 Localized r-robust code $\mathtt{rID-LOCAL}(r,G)$

We start with a graph $G=(V,E)$ and a non-negative integer r. The greedy set multicover approximation is denoted SET-MULTICOVER$(k,\mathbf{U},\mathcal{S})$.

```
1) Compute the set of nodes pairs, which are two hops
   away, $\bar{\mathbf{U}}$.
2) Compute $\bar{\Delta}=\{\bar{\delta}_u|u\in V\}$ using (1).
3) Apply $\mathbf{C}\leftarrow$SET-MULTICOVER$(2r+1,\bar{\mathbf{U}},\bar{\Delta})$
4) Output $\mathbb{C}_{local}\leftarrow\{u\in V|\bar{\delta}_u\in\mathbf{C}\}$
```

Theorem 2 *Given an undirected graph $G=(V,E)$ of n vertices, the performance ratio $\mathtt{rID-LOCAL}$ is upper bounded by:*

$$\frac{c_{greedy}}{c_{min}}<\ln\gamma+1,$$

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 [20], which is upper bounded by $1+\ln\alpha$ for a maximum set size α. The size of $\bar{\delta}_v$ is $|B(v)|(|B(v;3)|-|B(v)|+1)$, which, at its maximum, can be applied to the performance guarantee in [20] 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 2 can be significantly tighter.

In the next subsection we present a distributed implementation of the identifying code localized approximation. The following lemma supplements Lemma 2 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 3 *The distinguishing sets $\bar{\delta}_v$ and $\bar{\delta}_u$ are disjoint for every pair (u,v) with $\rho(u,v) > 4$.*

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

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

Moreover, for all such (x,y), $\rho(x,v) \leq 3$ and $\rho(y,v) \leq 3$. On the other hand, for $(x',y') \in \bar{\delta}_u$ with $\rho(u,v) > 4$, either x' or y' must be a neighbor of u, and hence of distance > 3 from v. Thus, $\bar{\delta}_v$ and $\bar{\delta}_u$ are disjoint. ∎

Lemma 3 implies that, when applying the greedy 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.

III. Distributed algorithms

Several parallel algorithms exist in the literature for set cover and for the more general covering integer programs (*e.g.,* [22]). 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.,* [23]). In a recent work, Kuhn et al. [24] devised a distributed algorithm for finding a dominating set with a constant runtime. The distributed algorithm uses a design parameter which provides a tradeoff between the runtime and performance.

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. The first is completely asynchronous, guarantees a performance ratio of at most $\ln\gamma + 1$, and requires $\Theta(c_{\text{dist}})$ iterations at worst,

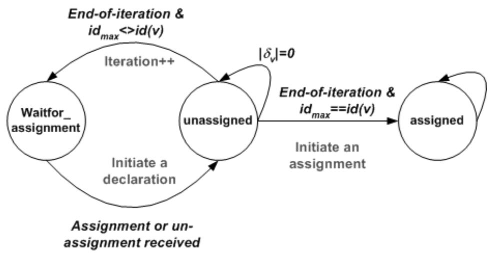


Fig. 4. Asynchronous distributed algorithm state diagram in node $v \in V$

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 $\ln\gamma + 1$, and for some arbitrarily small $\epsilon > 0$ operates within $O\left(\frac{\gamma n^{\frac{K+2+\epsilon}{K-1}}}{K}\right)$ time slots (resulting in $O(c_{\text{dist}} \max_{v\in V} |B(v;4)|)$ 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.

A. Setup and initialization

With a setup similar to [6] 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 $\bar{\delta}_v$ given in (2). This distinguishing set is updated as new codewords are added to the identifying code being constructed, $\mathbb{C}$; their presence is advertised by flooding their four-hop neighborhood.

B. The asynchronous algorithm $\mathtt{rID}-\mathtt{ASYNC}$

The state diagram of the asynchronous distributed algorithm is shown in Figure 4. 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 if 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 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 $|\bar{\delta}_v| = 0$ or in the "assigned" state. Clearly, nodes that terminate in the "assigned" state constitute a localized r-robust identifying code.

Algorithm 3 Asynchronous r-robust algorithm ($\mathtt{rID-ASYNC}$)

We start with a graph G, with vertices labeled by ID, and a non-negative integer r. The following distributed algorithm run at node $v \in V$ produces an r-robust identifying code.

```
Precomp    • Compute $\bar{\delta}_v$ using (2).
           • Initiate a declaration message and set $state = unassigned$.
           • Set $ID_{max} = ID(v)$, $\delta_{max} = |\bar{\delta}_v|$.
Iteration  • Increment $hop(ms)$ and forward all messages of $hop(ms) < 4$.
           • if received an assignment message with $state \neq assigned$ then
               – Update $\bar{\delta}_v$ by removing all pairs covered $2r+1$ times.
               – Initiate a declaration message and set $state = unassigned$.
               – Perform Comp($ID(v)$, $|\bar{\delta}_v|$).
           • if $state = waitfor-assignment$ and received an unassignment
             message then
               – Initiate a declaration message and set $state = unassigned$.
               – Perform Comp($ID(v)$, $|\bar{\delta}_v|$).
           • if received a declaration message $ms$ with $state \neq assigned$ then
             perform Comp($ID(ms), data(ms)$).
           • if end-of-iteration reached then,
               – if $ID_{max} = ID(v)$ and $|\bar{\delta}_v| > 0$ then $state = assigned$,
                 initiate an assignment message.
               – otherwise $ID_{max} = ID(v)$, $\delta_{max} = 0$, and state=waitfor-
                 assignment.
Comp($id$,•$\delta$) if $\delta_{max} < \delta$ or ($\delta_{max} = \delta$ and $ID_{max} > id$) then $\delta_{max} =$
           $\delta$, $ID_{max} = id$
```

1) Example: We illustrate the operation of $\mathtt{rID-ASYNC}(0, G)$ over a simple ring topology of 10 nodes in Figure 5. 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 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 5. In total $\mathtt{rID-ASYNC}$ returns an identifying code of size 6 - only one node more than a minimum one. The outcome of $\mathtt{rID-ASYNC}$ 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.

2) Performance evaluation:

Theorem 3 *The algorithm* $\mathtt{rID-ASYNC}$ *requires* $\Theta(c_{dist})$ *iterations and has a performance ratio*

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

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

The first part of the Theorem follows from Theorem 2 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 5) demonstrates that, in the worst case, exactly one node is assigned per iteration.

It follows that the amount of communications required in the iteration stage is $\Theta(c_{\mathrm{dist}}|V| \max(|B(v;4)|))$, 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

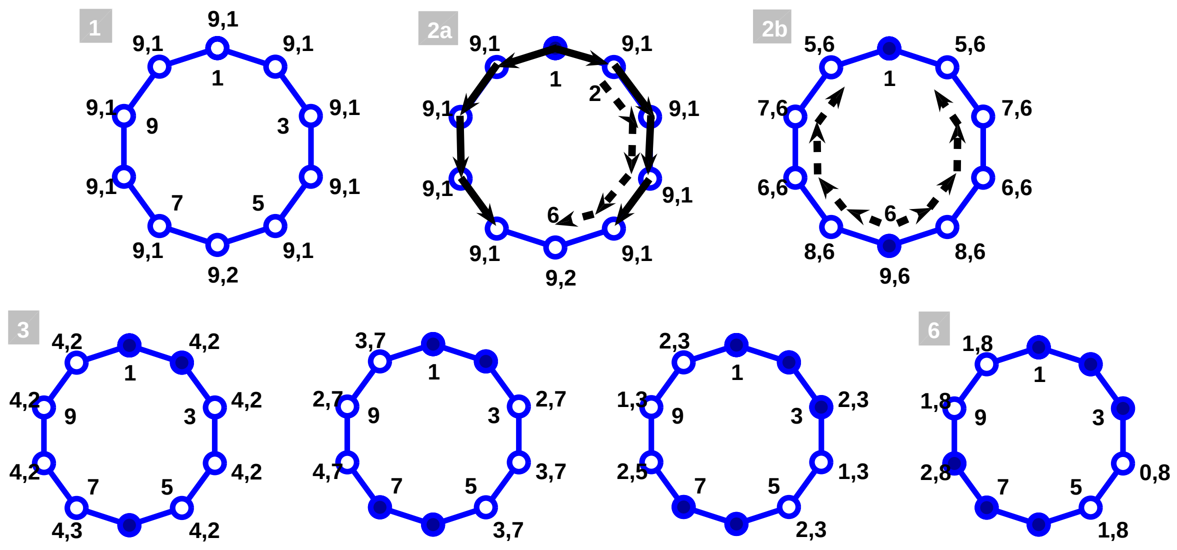


Fig. 5. Operation of $\texttt{rID}-\texttt{ASYNC}$ over a ring of 10 nodes, labeled from 1 to 10 clockwise (displayed in the inner perimeter), and $r = 0$. Solid circles represent assigned vertices, and the size of the distinguishing set and the value of ID_{max} at the end of each iteration appear in 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.

synchronized distributed algorithm that eliminates declaration messages altogether.

C. *A low-communications randomized algorithm* $\texttt{rID}-\texttt{SYNC}$

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 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.*, $F = \frac{n(n-1)}{2}$ will always work). A 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.

1) Iterations 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 4 Synchronous r-robust algorithm ($\texttt{rID}-\texttt{SYNC}$)

We start with a graph G and non-negative integer r. The following distributed algorithm run at node $v \in V$ produces an r-robust identifying code.

```
Precomp  • Set: $slot = s_F$, $subslot = L$, $state = unassigned$.
         • Compute $\bar{\delta}_v$ using (2) and set $as = |\bar{\delta}_v|$
Iterate: while $state = unassigned$ and $slot \geq s_1$ do,
         • $l = random\{1, ..., L\}$
         • Increment $hop(ms)$ and forward all messages of $hop(ms) < 4$.
         • if received assignment message, $ms$ then,
           update $\bar{\delta}_v$ by removing all pairs covered $2r+1$ times and set $as = |\bar{\delta}_v|$.
         • elseif $subslot = l$ and $slot = as$ then,
           Transmit an assignment message, $state = assigned$.
```

2) Example: We illustrate in Figure 6 the operation of $\texttt{rID}-\texttt{SYNC}(0, G)$, given in Algorithm 4, 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

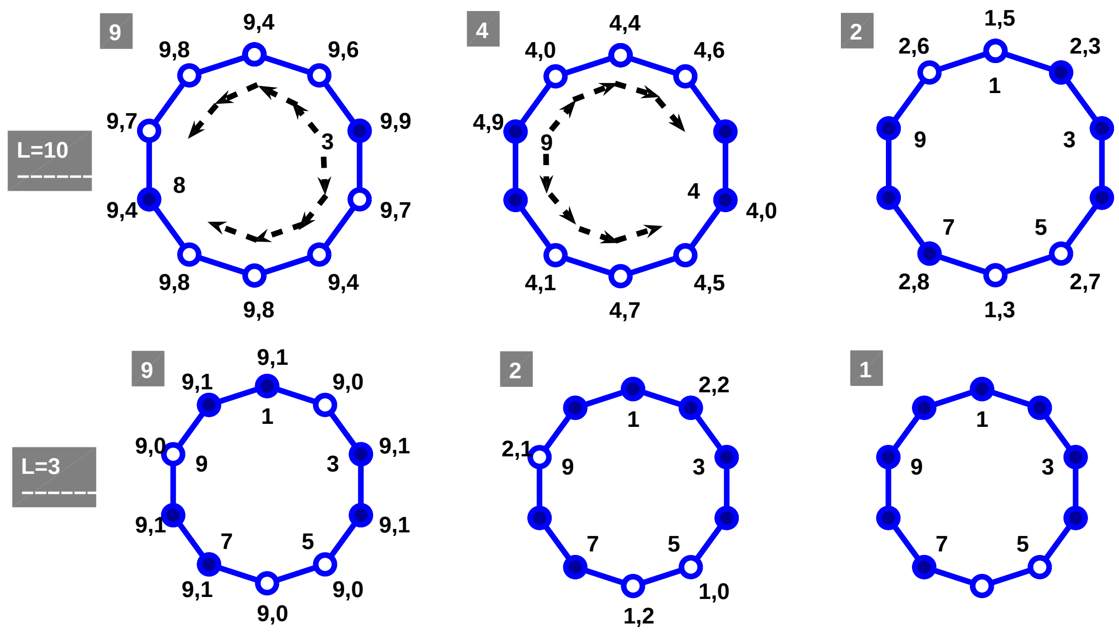


Fig. 6. Operation of $\texttt{rID}-\texttt{SYNC}$ over a ring of 10 nodes, labeled from 1 to 10 clockwise (displayed in the inner perimeter) for $r=0$ and $L=10$ (top) and $L=3$ (bottom). Solid circles represent assigned vertices, and the size of the distinguishing set and randomly selected subslot, 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.

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 $B(3;4)$ it is free to assign itself as indicated in the subfigure. Note this fundamental difference from $\texttt{rID}-\texttt{ASYNC}$ where in a similar situation node 8 wouldn't be assigned (see Figure 5). $\texttt{rID}-\texttt{SYNC}$ 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 $\texttt{rID}-\texttt{ASYNC}$ The outcome of $\texttt{rID}-\texttt{SYNC}$ can vary depending on the way nodes resolve ties and therefore is sensitive to the random selection of subslots. The bottom of Figure 6 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.

3) Performance evaluation: Algorithm $\texttt{rID}-\texttt{SYNC}$ 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.,* $O(c_{\text{dist}}\cdot\max_{v\in V}(|B(v;4)|))$), and includes only assignment messages, which are propagated to four hop neighborhoods.

In the case of ties, $\texttt{rID}-\texttt{SYNC}$ 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 4 *For asymptotically large graphs, Algorithm* $\texttt{rID}-\texttt{SYNC}$ *guarantees (with high probability) a performance ratio of*

$$\frac{c_{dist}}{c_{min}} < K(\ln\gamma+1),$$

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

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 2, as in the theorem statement. We next determine the number of subslots L needed to guarantee the above 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, $P(K)\geq(1-\bar{p}(K))^{c_{\text{dist}}}$, where $\bar{p}(K)$ 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

$$\begin{aligned}\bar{p}(K) &= \textstyle\sum_{k=K}^{t} L\binom{t}{k}L^{-k}\left(1-\frac{1}{L}\right)^{t-k} \leq \sum_{k=K}^{t}\binom{t}{k}L^{1-k} \\ &\leq \textstyle\sum_{k=K}^{t} L\left(\frac{te}{Lk}\right)^{k} \leq tL\left(\frac{te}{LK}\right)^{K},\end{aligned}$$

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 $L = \frac{e}{K}n^{\frac{K+2+\epsilon}{K-1}}$. Then,

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

■

IV. Random graphs

Recently, Erdos-Renyi random graphs and random geometric graphs were studied in the context of identifying codes [16, 25]. In [16] it was shown that for asymptotically large random graphs, any subset of a certain threshold size (logarithmic in the size of the graph) is almost surely an identifying code. It was also shown that the threshold is asymptotically sharp, *i.e.*, the probability of finding an identifying code of slightly smaller size asymptotically approaches zero.Unit disk geometric random graphs, in which vertices are placed on a two-dimensional plane and connected if their distance is less than some unit, were studied in [25]. Unlike large Erdos-Renyi random graphs, most of the large unit-disk geometric random graphs do not possess identifying codes.

The study of random graphs is important since it can offer insights and provide designs of such codes. In this section we extend the work in [16] to provide a threshold on the size of an r-robust identifying code in asymptotically large Erdos-Renyi random graphs.

Let $G(n,p) = (V,E)$ be a random graph of n vertices where the existence of an edge between any pair of distinct vertices is determined independently and in random with probability p. The following theorem hold asymptotically in n.

Theorem 5 *Let $r \leq O(\log\log\log n)$, and $q = p^2 + (1-p)^2$ such that $1 - q \geq \Omega\left(\frac{\log n}{n}\right)$. Then asymptotically in n and for an arbitrary small $\epsilon > 0$ any subset of size*

$$c \geq \frac{2\log n + (2r-1+\epsilon)\log\log n}{\log 1/q}$$

is almost surely an r-robust identifying code.

The proof appears in the Appendix.

Recall that Moncel *et al.* showed in [16] that any subset of size $\frac{(2+\epsilon)\log n}{\log 1/q}$ is almost surely an identifying code. Theorem 5 implies that for a small r a relatively small addition of $(2r-1)\log\log n$ vertices almost surely makes the code r-robust. Theorem 5 also provide means of random design of r-robust identifying codes. Figure 7 shows a numerically derived upper

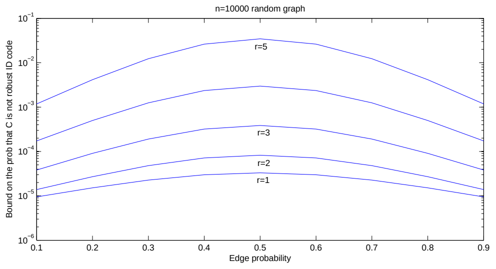


Fig. 7. An upper bound on the probability that a set of vertices of the size of Theorem 5 is not a robust identifying code

bound on the probability that a set of the size of Theorem 5 is *not* r-robust, for random graph of 10000 vertices and $r \leq 5$.

In [16] it was also shown that the threshold size of an identifying code is tight in the sense that all slightly smaller subsets are almost surely not identifying codes. Clearly, this result also holds for robust identifying codes. It is not obvious if this bound can be tightened for robust identifying codes.

V. Routing with identifying codes

The existence of an identifying code for a network permits a natural routing scheme using small routing tables (typically referred to as “compact routing schemes” [26–28]).

By routing scheme we mean a combination of a labeling scheme L (consisting of a label/address L_i for node i), routing scheme T (consisting of a routing table T_i at node i), and a routing function $f(L_s, L_t, L_i, T_i)$. The routing table uses the labels of the source node s, the destination node t, the current node i, and the information in the local routing table T_i to choose the next port through which a packet should be sent. For the scheme to be considered *compact*, the table size should be small (*i.e.*, $\forall i\, |T_i| \ll O(N)$), and the label size should also be small (usually polylogarithmic in N). Furthermore, the description of f should be of constant size (*e.g.*, we do not want to include the whole graph structure in f) and its time complexity should be low (usually polynomial in label size and logarithmic or constant in the table size).

Compact routing has been studied for some time in the computer science literature, with the typical focus being on designing routing schemes that give good performance in the worst case scenario for all graphs, or for some class of graphs; a good survey of existing approaches is provided by [29]. The closest related routing scheme in the literature is based on a dominating set, and this is not surprising because identifying codes are a special type of dominating set.

A. Related Work

Considerable work (see e.g., [30–32]) has been done on compact routing using dominating sets, and, in particular connected dominating sets. In the context of ad hoc and sensor networks these methods are referred to as clustering algorithms (see e.g., [33, 34] for a survey). Dominating sets are sets of nodes (also called clusterheads) whose combined neighborhoods (clusters) include all nodes in the graph. Therefore, if information for the shortest path routing to each clusterhead (node in a dominating set) is stored, any node can be reached through one of its clusterheads with an almost optimal path length. In connected dominating sets [30, 32] the routing is done only using nodes from the dominating set, which serve as the network's "backbone". This has the advantage that other nodes can be added and removed from the network without affecting the routing, but it also has the disadvantage of possibly considerably lengthening the routing distance. Most of the work on (connected) dominating sets based routing has focused on efficiently finding small (connected) dominating sets, commonly resulting in little overlap between clusters, *i.e.,* small amount of nodes that belong to more than one cluserhead. Although minimizing the clusterhead infrastructure size is in general desirable it poses robustness issues, as a single cluserhead failure most likely results in a disconnected topology. To reduce this problem denser clusterhead infrastructures that provide redundancy in the event of a failure were suggested (see *e.g.,* [34] for a survey).

The routing scheme presented here is based on identifying codes (the clusterheads form an identifying code in the graph representing the network), which generally form a superset of a dominating set. The proposed scheme sports the following advantages:

- It combines the routing and the monitoring infrastructures; hence saving both energy and setup time.
- It provides a natural labeling scheme for the network by simply using the identifying sets of the code. Dominating-set routing schemes typically assume that such a labeling is externally determined.
- It provides robustness against failures of a limited amount of infrastructure nodes, as commonly nodes are covered by more than one codeword (clusterhead).
- Its routing length is comparable to dominating set routing schemes.

The identifying codes based routing scheme suggested here guarantees a near optimum routing length, but uses non code nodes for routing. Our method can be modified to utilize only nodes in some connected identifying set by using *dynamic* identifying sets, which are basically identifying codes that form a walk in the graph, and therefore a connected set of nodes (see [35]). In some cases the size of an identifying code may be close to the size of a dominating set. For example, in a random graph, the identifying code is only larger than a dominating set by at most a logarithmic factor.

Still, in general the size of an identifying code will be larger than the size of a dominating set. Moreover, identifying codes may not exist for some graphs (though dominating sets always exist). In such a case our proposed algorithms permit a small number of nodes to share the same identifying sets (or label) - minimally breaking the identifying property of the code. The nodes that break the identifying property are distinguished by other means, such as adding distinguishing bits to their label. The maximum number of added bits in an efficient scheme is approximately $\log_2$ of the size of the largest set of indistinguishable nodes. Since the indistinguishable nodes are guaranteed to be within 2 hops of each other, this task becomes relatively simple.

B. Routing with an identifying code

Given identifying code for a graph G, our scheme induces a compact routing scheme as follows: Number the codewrods in $\mathbb{C}$ as $c_0, \ldots, c_{|\mathbb{C}|-1}$; the label of node i will be the characteristic vector of its identifying set (and is thus unique). At every node, the routing table will include one entry for each of the codewords, which will include the port leading to the shortest path to this codeword.

The routing function f at some node i will be as follows:

1) If t is i's neighbor, send directly to t.
2) Otherwise, choose a codeword c_j, such that $c_j \in B(i)$, *i.e.,* such that the j-th bit of L_i is one. Route by the port assigned to c_j in the routing table T_i.

We note that the routing scheme presented here may be extended to a hierarchical routing scheme using higher radius identifying codes to further reduce the size of the routing table [36].

For a graph permitting an identifying code, we can see that the routing table size is at most $|\mathbb{C}|^2$ bits, the label size is $|\mathbb{C}|$, and the routing function runs in time linear in the label size. If $|\mathbb{C}|$ is large but the size of $I_{\mathbb{C}}(u)$ is small for all $u \in V$, a more compact label may be used by choosing a different representation of the list: either a linked list of codewords or a run length encoding of the label.

Theorem 6 *The function f is a valid routing function.*

Proof: At every node the routing table includes an entry for each of the codewords. The entry contains the next port in the shortest path routing to the codeword. Therefore, shortest path routing to the selected codeword is guaranteed. Since the selected codeword is a neighbor of the destination, the packet will be directly routed once the codeword is reached.

■

Interestingly, the routing distance $r(s,t)$ between nodes s and t is almost identical to the shortest path distance $d(s,t)$.

Theorem 7 *The routing scheme above guarantees that* $r(s,t) \leq d(s,t) + 2$.

Proof: If t is a codeword then routing to t is done using the shortest path by the routing tables. Suppose t is not a codeword, and assume c is in the identifying set of t. The routing scheme routes to c by shortest path, and then to t by one more hop. Therefore $r(s,t) \leq d(s,c) + 1$. By the triangle inequality $d(s,c) \leq d(s,t) + d(t,c) = d(s,t) + 1$. The theorem follows. ■

The possibility of routing using the codewords is based on the code also being a dominating set. The creation of

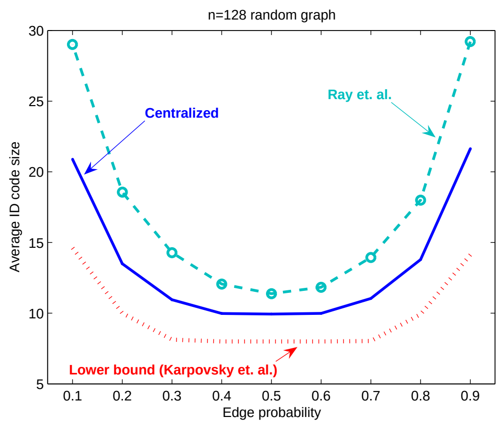


Fig. 8. Average simple identifying code ($r = 0$) size of the proposed $\mathtt{rID-CENTRAL}$ and ID-CODE algorithms for 128 nodes random graphs with different edge probabilities, in comparison to a lower bound of [9].

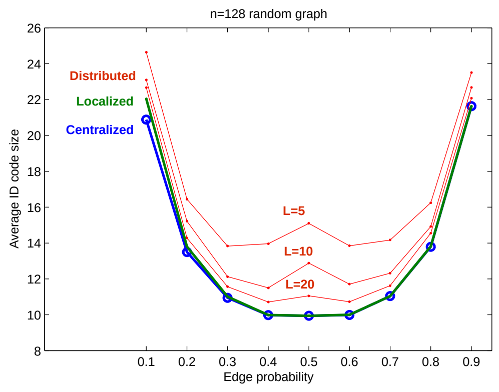


Fig. 10. Average simple identifying code ($r = 0$) size of the proposed $\mathtt{rID-CENTRAL}$, $\mathtt{rID-LOCAL}$, and $\mathtt{rID-SYNC}$with different number of subslots parameter, L, for 128 nodes random graphs with different edge probabilities.

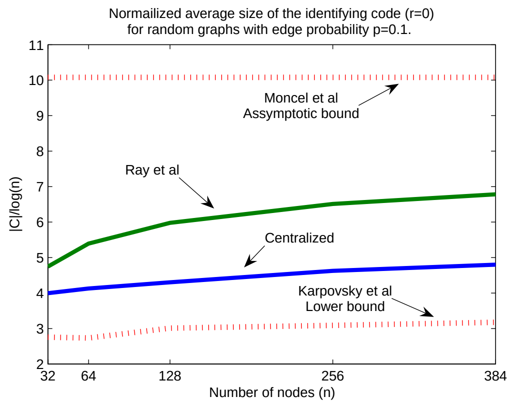


Fig. 9. Average size of the simple identifying code ($r = 0$) for random graphs with edge probability $p = 0.1$, and various numbers of vertices.

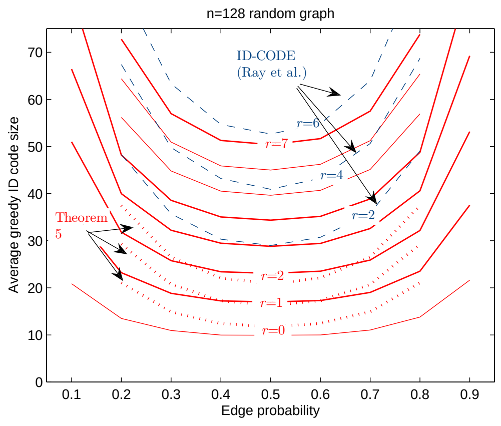


Fig. 11. Centralized r-robust identifying codes algorithm (solid), ID-CODE algorithm (dashed), and the asymptotic bound of Theorem 5 (dotted) for 128 nodes random graphs with different edge probabilities.

an identifying set for identification purposes permits the use of this set in a natural way to achieve compact routing. The usage of the identifying code, rather than a possibly smaller dominating set, has the advantage of labeling the nodes in a natural way, requiring only an *a priori* agreement on the labels of codewords (rather than all the nodes in the graph). It also provides a higher robustness in the event of infrastructure nodes failures. In addition it also permits the distributed construction of the labeling scheme and routing tables based on the distributed identifying code algorithms presented earlier.

VI. Simulations

We have simulated the centralized ($\mathtt{rID-CENTRAL}$), localized ($\mathtt{rID-LOCAL}$) and distributed asynchronous and synchronous ($\mathtt{rID-ASYNC}$, $\mathtt{rID-SYNC}$) identifying code algorithms, and applied them to random graphs with different edge probabilities, and to geometric random graphs with different nodes densities. We have used the averaged size of the identifying code as a performance measure.

For the case of $r = 0$ (*i.e.*, simple identifying code) the simulation results are compared to ID-CODE, the algorithm suggested by Ray et. al. in [6]. In addition, our figures mark the combinatorial lower bound first derived by Karpovsky et. al. in [9], and the asymptotic result (in n - the size of the graph) of Moncel et. al. [16], who showed that an arbitrary collection of an (asymptotically tight) threshold number of codewords is an identifying code (with high probability).

Figure 8 compares our centralized greedy algorithm to ID-CODE and the combinatorial lower bound, with our algorithm demonstrating a significant improvement over ID-CODE. It should be noted that as n grows, the curves for basically any algorithm should converge very slowly to Moncel's asymptotic result, as illustrated in Figure 9. This apparently slow convergence rate suggests that there is a lot to gain from using the suggested algorithms, even for reasonably large networks [16].

Figure 10 shows the simulation results for the localized and distributed algorithms compared to the centralized one. Recall that the performance of the asynchronous algorithm, $\mathtt{rID-ASYNC}$, is identical to the localized approximation, and

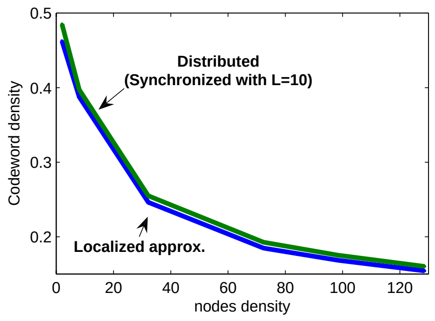


Fig. 12. Codeword density (the size of the code normalized to a unit area) for the localized ($\mathtt{rID}-\mathtt{LOCAL}$) and distributed ($\mathtt{rID}-\mathtt{SYNC}$) algorithms for GRGs with different nodes densities (nodes per unit area).

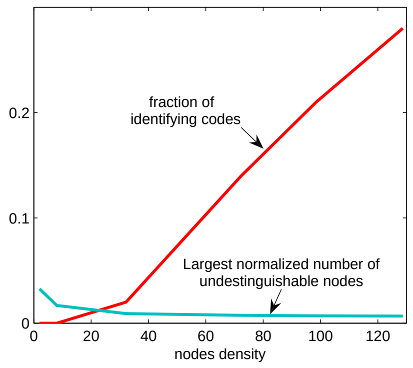


Fig. 13. Fraction of graphs admitting an identifying code, and maximum fraction of indistinguishable nodes for GRGs with different node densities (nodes per unit area).

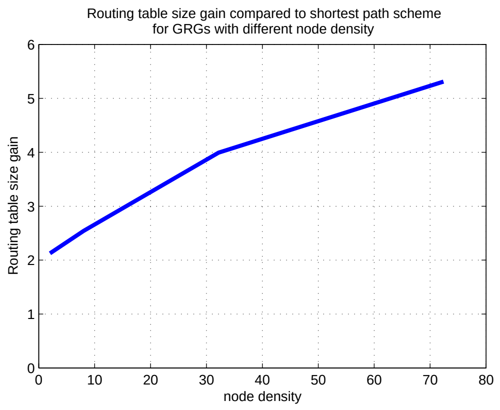


Fig. 14. Ratio of the graph size to the size of the routing table using the identifying code approach. This represents the ratio of the full routing table containing all nodes to the more compact routing table presented here. Results are for geometric random graphs with different node densities.

the simulation results of the localized algorithm nearly match the results of the centralized algorithms. 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, $\mathtt{rID}-\mathtt{SYNC}$. 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. The results of the centralized r-robust identifying code algorithm are shown in Figure 11 versus the theoretical results of Theorem 5, and the results of the ID-CODE algorithm [6]. As in the simple ($r = 0$) identifying code case, our algorithm outperforms the ID-CODE algorithm for a small and moderate r.

Figure 12, 13 show the codeword density for geometric random graphs 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 approaches (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. However, the monitoring and routing functionality can still be restored by a special treatment of a small fraction of indistinguishable nodes.

Finally, Figure 14 presents the ratio of the size of the full routing table containing all nodes to the random table containing only codewords for geometric random graphs. In cases where no identifying code exists, information on indistinguishable nodes was added to the routing table. The table size for our identifying code based routing is clearly much smaller than for the full routing table.

VII. Conclusions and further study

In this paper we have proposed a new polynomial-time approximation algorithm for the minimum r-robust identifying code problem with provable performance guarantees. Our algorithm generates a robust identifying code whose size is guaranteed to be at most $1 + 2\log(n)$ times larger than the optimum, where n is the number of vertices. To the best of our knowledge, this is the first such approximation in the literature. We have also proposed two distributed implementations of our algorithm. The first implementation provides a tradeoff between runtime and performance guarantees, while using a low communications load and only coarse synchronization; the second implementation requires no synchronization at the expense of more communication. Through simulations on Erdos-Renyi random graphs and geometric random graphs, we have shown that these algorithms significantly outperform earlier identifying code algorithms.

Finally, we have demonstrated that the same identifying code-based monitoring infrastructure can be reused to efficiently implement routing between any two nodes in the network. Specifically, we have shown how to use routing tables of the same size as the network's identifying code to route packets between any two nodes within two hops of the shortest path. The significance of this result is two-fold: (i) we can perform near-optimal routing while significantly compressing routing table memory in each sensor; (ii) one algorithm can be run to simultaneously setup both monitoring and routing infrastructures, thus reducing the network setup overhead.

r-Robust identifying codes provide means of robustness to the monitoring infrastructure in a straightforward sense: areas continue to be monitored even if any subset of r monitors malfunction. It seems that this robustness property can be beneficial for the routing infrastructure as well; for example, it can simplify or completely eliminate the rearrangement of routing infrastructure during routers failures. Still, such routing protocols are not trivial and should be carefully studied.

VIII. Acknowledgements

This material is based, in part, upon work supported by the National Science Foundation under Grants 0132802, CCF-0729158, CCR-0133521 and CNS-0435312.

References

[1] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet," in *ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, 2002, pp. 96–107.

[2] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong, "A macroscope in the redwoods," in *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, 2005, pp. 51–63.

[3] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A wireless sensor network for structural monitoring," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004.

[4] "Pump monitoring at a nuclear generating station," 2005, sensicast Systems, Inc. [Online]. Available: http://www.sensicast.com/solutions/casestudys.php

[5] S. Ray, R. Ungrangsi, F. D. Pellegrinin, A. Trachtenberg, and D. Starobinski, "Robust location detection in emergency sensor networks," *Proceedings INFOCOM*, p. 10441053, April 2003.

[6] S. Ray, D. Starobinski, A. Trachtenberg, and R. Ungrangsi, "Robust location detection with sensor networks," *IEEE Journal on Selected Areas in Communications (Special Issue on Fundamental Performance Limits of Wireless Sensor Networks)*, vol. 22, no. 6, August 2004.

[7] T. Berger-Wolf, W. Hart, and J. Saia, "Discrete sensor placement problems in distribution networks," *SIAM Conference on Mathematics for Industry*, October 2003.

[8] ——, "Discrete sensor placement problems in distribution networks," *Journal of Mathematical and Computer Modelling*, vol. 42, no. 13, 2005.

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

[10] 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, 2003.

[11] ——, "Identifying and locating-dominating codes: NP-completeness results for directed graphs," *IEEE Transactions on Information Theory*, vol. 48, no. 8, pp. 2192–2200, August 2002.

[12] J. Moncel, "On graphs of n vertices having an identifying code of cardinality $\lceil log_2(n+1)\rceil$," *Discrete Applied Mathematics*, vol. 154, no. 14, pp. 2032–2039, 2006.

[13] M. Laifenfeld, "Localization and identification in networks using robust identifying codes," *Information Theory and Application Workshop*, 2008.

[14] M. Laifenfeld and A. Trachtenberg, "Identifying codes and covering problems," *To appear in IEEE Transaction on Information Theory, Sept. 2008.*

[15] M. Laifenfeld, A. Trachtenberg, R. Cohen, and D. Starobinski, "Joint monitoring and routing in wireless sensor networks using robust identifying codes," *IEEE Broadnets 2007*, pp. 197 – 206, September 2007.

[16] J. Moncel, A. Frieze, R. Martin, M. Ruszink, and C. Smyth, "Identifying codes in random networks," *IEEE International Symposium on Information Theory, Adelaide, 4-9 Sept.*, 2005.

[17] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 2001.

[18] D. S. Johnson, "Approximation algorithms for combinatorial problems," *Journal of Computer and System Sciences*, vol. 9, pp. 256–278, 1974.

[19] U. Feige, "A threshold of ln n for approximating set cover," *Journal of the ACM*, vol. 45, no. 4, pp. 634–652, 1998.

[20] V. Vazirani, *Approximation Algorithms*. Springer-Verlag, July 2001.

[21] M. Laifenfeld and A. Trachtenberg, "Disjoint identifying codes for arbitrary graphs," *IEEE International Symposium on Information Theory, Adelaide, Australia*, 4-9 Sept 2005.

[22] S. Rajagopalan and V. Vazirani, "Primal-dual RNC approximation algorithms for set cover and covering integer programs," *SIAM Journal on Computing*, vol. 28, pp. 525–540, 1998.

[23] Y. Bartal, J. W. Byers, and D. Raz, "Global optimization using local information with applications to flow control," in *IEEE Symposium on Foundations of Computer Science*, October 1997, pp. 303–312. [Online]. Available: citeseer.ist.psu.edu/bartal97global.html

[24] F. Kuhn and R. Wattenhofer, "Constant-time distributed dominating set approximation," *Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing (PODC'03)*, pp. 25–32, July 2003. [Online]. Available: citeseer.ist.psu.edu/kuhn03constanttime.html

[25] T. Müller and J.-S. Sereni, "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.

[26] D. Peleg and E. Upfal, "A tradeoff between space and efficiency for routing tables," *J. ACM*, vol. 36, pp. 510–530, 1989.

[27] M. Thorup and U. Zwick, "Compact routing schemes," in *Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures*. ACM Press, 2001, pp. 1–10.

[28] L. Cowen, "Compact routing with minimum stretch," *Journal of Algorithms*, vol. 38, no. 1, pp. 170–183, 2001.

[29] C. Gavoille, "A survey on interval routing schemes," *Theoret. Comput. Sci.*, vol. 249, pp. 217–253, 1999.

[30] A. T. Jeremy Blum, Min Ding and X. Cheng, in *Handbook of Combinatorial Optimization*, D.-Z. Du and P. Pardalos, Eds. Kluwer Academic Publishers, 2004, ch. Connected Dominating Set in Sensor Networks and MANETs.

[31] J. Wu and H. Li, "A dominating-set-based routing scheme in ad hoc wireless networks," *Telecommunication Systems*, vol. 18, pp. 13–36, 2001.

[32] J. Wu, in *Handbook of Wireless Networks and Mobile Computing*, I. Stojmenovic, Ed. Wiley, 2002, ch. Dominating-Set-Based Routing in Ad Hoc Wireless Networks.

[33] Y. Chen, A. Liestman, and J. Liu, "Clustering algorithms for ad hoc wireless networks," *In Ad Hoc and Sensor Networks*, 2004. [Online]. Available: citeseer.ist.psu.edu/chen04clustering.html

[34] J. Yu and P. Chong, "A survey of clustering schemes for mobile ad hoc networks," *IEEE Communications Surveys and Tutorials*, vol. 7, pp. 32–48, 2005.

[35] I. Honkala, M. Karpovsky, and L. Levitin, "On robust and dynamic identifying codes," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 599–612, February 2006.

[36] I. Abraham and D. Malkhi, "Compact routing on euclidean metrics," in *23rd ACM Symposium on Principles of Distributed Computing (PODC 2004)*. ACM Press, 2004.

IX. Appendix

We provide here the proof of Theorem 5. Recall that $G(n,p) = (V,E)$ is a random graph of n vertices where the existence of an edge between any pair of distinct vertices is determined independently and in random with probability p.

Proof of Theorem 5: The proof is based on developing the probability that a subset of size c is *not* a robust identifying code and showing that this probability goes to zero asymptotically in n. But first we show that the expression for c can always be made feasible under the settings of the theorem, namely that $c \leq n$.

Observe that $\log 1/q = \log(1 + (1/q - 1)) \geq (1/q - 1) - \frac{1}{2}(1/q-1)^2 = (1/q-1)[1-\frac{1}{2}(1/q-1)] \geq \frac{1}{2}(1/q-1)$, where the last inequality follows from the fact that $0.5 \leq q \leq 1$. If $r \leq O(\log\log\log n)$ then the bound on c can be upper bounded by

$$\begin{aligned}
\frac{2\log n + 2r\log\log n}{\log 1/q} &\leq \frac{4q\left(\log n + r\log\log n\right)}{1-q} \\
&\leq \frac{4q\left(\log n + O(\log\log\log n)\log\log n\right)}{1-q} \\
&\leq \frac{O(\log n)}{1-q}
\end{aligned}$$

Since $1 - q \geq \Omega\left(\frac{\log n}{n}\right)$, it follows that there exist constants $K > K_0 > 0$ such that for large enough n, $1 - q \geq \frac{K_0 \log n}{n}$, and for large enough K our assertion that $c \leq n$ can be made true.

We next show that any subset of size c is almost surely an r-robust identifying code. Fix $\mathbb{C}$ to be a code of size c and let $q = p^2 + (1-p)^2$ be the probability that a codeword in $\mathbb{C}$ does not *distinguish* between a pair of vertices in V, namely the probability that the codeword doesn't appear in the pair's difference set. In the rest of the proof we develop an upper bound on the probability that $\mathbb{C}$ is not an r-robust identifying code and show that for a certain size c it goes to zero asymptotically in n.

By Lemma 1 if $\mathbb{C}$ is not r-robust then there must be a pair of distinct vertices $u, v \in V$, for which the difference set under $\mathbb{C}$ has at most $2r$ elements. The probability of such an event, $Pe(u,v)$, depends whether either u or v, or both are in $\mathbb{C}$:

1) Both u, v are in $\mathbb{C}$. There are two possibilities here that need to be addressed separately; either there is an edge between u and v, and hence there can be at most $2r$ distinguishing codewords from the remaining $c-2$ codewords, namely $Pe(u,v|u,v \in \mathbb{C}, (u,v) \in E) = p\sum_{i=0}^{2r}\binom{c-2}{i}(1-q)^i q^{c-2-i}$, or there is no edge between u and v, implying that both u and v are already in the distinguishing set; therefore there can be at most $2r-2$ additional distinguishing codewords from the remaining $c-2$ codewords, namely $Pe(u,v|u,v \in \mathbb{C}, (u,v) \notin E) = (1-p)\sum_{i=0}^{2r-2}\binom{c-2}{i}(1-q)^i q^{c-2-i}$.
   In total we have:
   $$\begin{aligned}
   Pe(u,v|u,v \in \mathbb{C}) = &\textstyle\sum_{i=0}^{2r-2}\binom{c-2}{i}(1-q)^i q^{c-2-i} \\
   &\textstyle+p\sum_{i=2r+1}^{2r}\binom{c-2}{i}(1-q)^i q^{c-2-i}.
   \end{aligned}$$
2) Exactly one of u, v is in $\mathbb{C}$. Similarly to the previous case there are two possibilities here depending whether an edge between the two vertices exists. In total the probability Pe is given by
   $$\begin{aligned}
   Pe(u,v||\{u,v\} \cap \mathbb{C}| = 1) = &\textstyle\sum_{i=0}^{2r-1}\binom{c-1}{i}(1-q)^i q^{c-1-i} \\
   &\textstyle+p\binom{c-1}{2r}(1-q)^{2r} q^{c-1-2r}.
   \end{aligned}$$
3) Neither u nor v are in $\mathbb{C}$. Here there are at most $2r$ distinguishing codewords in $\mathbb{C}$.
   $$Pe(u,v|u,v \notin \mathbb{C}) = \sum_{i=0}^{2r}\binom{c}{i}(1-q)^i q^{c-i}.$$

We use the union bound to set an upper bound on the probability that $\mathbb{C}$ is not an r-robust identifying code:

$$\begin{aligned}
P(\mathbb{C}\text{ is not }r\text{-robust}) &\leq \binom{c}{2}Pe(u,v|u,v \in \mathbb{C}) + \\
&\quad (n-c)cPe(u,v||\{u,v\} \cap \mathbb{C}| = 1) + \\
&\quad \binom{n-c}{2}Pe(u,v|u,v \notin \mathbb{C}) \\
&\leq 3n^2 q^{c-2}\sum_{i=0}^{2r}\binom{c}{i}\left(\frac{1-q}{q}\right)^i \\
&\leq 3n^2 q^{c-2}\frac{\left(\frac{c(1-q)}{q}\right)^{2r} - 1}{\frac{c(1-q)}{q} - 1} \\
&\leq 3n^2 q^{c-2}\frac{\left(\frac{c(1-q)}{q}\right)^{2r}}{\frac{c(1-q)}{q} - 1}.
\end{aligned}$$

Next consider a code $\mathbb{C}$ of size $c = \frac{2\log n + (2r-1+\epsilon)\log\log n}{\log 1/q}$ for arbitrary small $\epsilon > 0$, and $r = O(\log\log\log n)$.

Under this selection of parameters we show that the following equations holds:

$$\begin{aligned}
q^c &= n^{-2}(\log n)^{-2r+1-\epsilon} && (3) \\
c^{2r} &\leq O\left(\left(\frac{2}{\log(1/q)}\right)^{2r}(\log n)^{2r}\right) && (4) \\
\frac{1-q}{\log 1/q} &= \Theta(1) && (5) \\
c(1-q) &\geq \Omega(\log n). && (6)
\end{aligned}$$

Equation (3) is straightforward.

To see why Equation (4) is true observe that

$$\begin{aligned}
c^{2r} &= \left(\frac{2}{\log(1/q)}\right)^{2r}(\log n)^{2r}\left(1 + O\left(\frac{\log\log n}{\log n}\right)\right)^{2r} \\
&\leq \left(\frac{2}{\log(1/q)}\right)^{2r}(\log n)^{2r}e^{O(1)}.
\end{aligned}$$

Equation (5) follows form the fact that $0.5 \leq q \leq 1$ and hence $\frac{0.5}{\log 2} \leq \frac{1-q}{\log 1/q} = \frac{1-q}{\sum_{i=1}\frac{(1-q)^i}{i}} = \frac{1}{1+\sum_{i=1}\frac{(1-q)^i}{i+1}} \leq 1$.

Finally Equation (6) follows from a directs substitution of c in Equation (5).

We can use these equations to complete the proof by further bounding the probability that $\mathbb{C}$ is not an r-robust identifying code:

$$\begin{aligned}
P(\mathbb{C}\text{ is not }r\text{-robust}) &\leq O\left((\log n)^{1-\epsilon}\frac{\left(\frac{2(1-q)}{q\log(1/q)}\right)^{2r}}{cq(1-q) - q^2}\right) \\
&\leq (\log n)^{1-\epsilon}\frac{(O(1))^{2r}}{\Omega(\log n)} \\
&\leq O\left(\frac{\log\log n}{(\log n)^{\epsilon}}\right) = o(1).
\end{aligned}$$

∎