Asynchronous and Noncoherent Neighbor Discovery for the IoT Using Sparse-Graph Codes

Kabir Chandrasekher[1], Kangwook Lee[2], Peter Kairouz[3], Ramtin Pedarsani[4], and Kannan Ramchandran[1]

[1]Electrical Engineering and Computer Science, University of California, Berkeley
[2]Electrical Engineering, KAIST
[3]Electrical Engineering, Stanford University
[4]Electrical and Computer Engineering, University of California, Santa Barbara

***Abstract*—In this paper, we design a fast and efficient energy-based and asynchronous neighbor discovery protocol for the Internet of Things (IoT). In our solution, we relax the assumption of frame-level synchronization. We formulate a novel asynchronous group testing scheme and apply it to the neighbor discovery problem. We then show that our proposed scheme is able to detect the set of K active neighbors[1] among a network of n nodes with codeword length and decoding complexity of $\Theta(K\log(K)\log(n))$. Finally, we provide extensive simulation results to verify our theoretical guarantees.**

I. Introduction

Network self-organization is becoming increasingly important with the emergence of the Internet of Things (IoT). In an IoT setting, there is an explosion of devices capable of collecting data and transmitting information. Many problems posed by this network of devices such as service recovery and energy management depend heavily on the knowledge of the local connectivity structure. For this reason, it is important that each device, henceforth referred to as a node, has knowledge of its adjacent nodes with which it can communicate [1]. Thus, often *the neighbor discovery phase*, a procedure during which nodes determine their neighboring nodes, precedes network self-organization. Traditional cellular random access schemes which are presently used to discover new devices do not scale to networks with potentially thousands of nodes per cell [2]. We propose a noncoherent, group testing based framework to bypass the need for large-scale channel estimation.

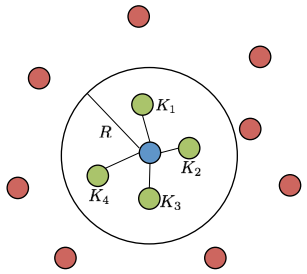


Fig. 1: **Spatial layout of neighbor discovery.** Consider the green nodes within a radius R of the ith node to be "active" among all of the nodes. K_i is the label of the ith active neighbor.

Multiple access and the neighborhood discovery problem have been studied in the literature from the group testing perspective [4], [5]. However, most research efforts have so far assumed that the nodes are tightly synchronized relative to each other and the base station prior to transmission. This assumption is rarely fulfilled in practice. The cost of synchronization is particularly significant in the IoT setting because the nodes are ultra-small, ultra-energy efficient and hence do not have access to a precise crystal oscillator. They instead have to rely on inaccurate and noisy clocks [6]. Furthermore, in the 60 GHz band, propagation delays can lead to significant frame level asynchronism even when the nodes are only tens of centimeters apart. Therefore, in this work, we relax the assumption that transmissions coming from various nodes are perfectly aligned and allow for frame level asynchronicity.

Of the existing works on asynchronous neighbor discovery, much effort has gone into designing protocols with energy-efficient schedules for duty cycles [7], [8]. Our work investigates a different problem, attempting to minimize discovery phase, assuming nodes are awake. Additionally, whereas there have been other works studying the problem from a random access point of view [9], [10], we focus primarily on efficient protocols utilizing a key assumption that the number of active neighbors can be sparse [11], and apply our proposed asynchronous group testing framework to solve the problem. We additionally note that sparse graph codes have been previously applied in the neighbor discovery and multiple access settings, however these works do not view the problem through the group testing point of view [12], [13].

The main contribution of the paper is as follows: We make essential use of the recently proposed SAFFRON (**S**parse-gr**A**ph codes **F**ramework **F**or g**RO**up testi**N**g) [14] algorithm for group testing, and robustify it to asynchronicity of the codewords, while maintaining its sample complexity and decoding complexity. Assuming that the maximum offset between the transmissions of the nodes is Δ, we show that under some mild assumptions, using asynchronous SAFFRON, a node is able to recover all its active neighbors with a communication overhead of $\Delta + cK\log(K)\log(n)$ bits, where c is a constant that can be precisely characterized.

The outline of the paper is as follows:

- In Section II, we present the problem formulation, describing the noncoherent neighbor discovery as group testing.
- In Section III, we briefly review the results and methods of the SAFFRON scheme and apply it to the neighbor discovery problem.
- In Section IV, we extend the SAFFRON scheme to asynchrony at the bit level. We robustify the Asyn-

[1]We note that while our scheme is designed for exact knowledge of K, it suffices to use an upper bound on K which can be estimated beforehand

chronous SAFFRON scheme to noise and provide extensive simulation results.

We also summarize the notation used in the paper as follows:

TABLE I: Notation

Notation	Description
n	Total number of nodes in the network
K	Total number of active users in range
m	Codeword length
M	Number of right nodes
Δ	Maximum delay of any node in the system
δ_i	Delay of node i
c_k	m-length codeword of user k
U	Signature matrix
$\mathbf{b}_i$	The $\lceil \log_2 n \rceil$-bits representation of i
$[N]$	$\{1, 2, \ldots, N\}$

II. Neighbor Discovery as Group Testing

In this section, we formulate the problem of detecting K active neighbors among n total nodes in the network through m bits of communication. We first consider the synchronous case where each active neighbor broadcasts a binary codeword of length m bits while each node in the network listens for the discovery period, assuming that the codewords are perfectly aligned. We then consider a more realistic scenario, which we call asynchronous neighbor discovery, where the transmissions of nodes are delayed, and the delays are unknown at the receiver.

A. Synchronous Neighbor Discovery

Let n be the number of nodes in the network (or devices in the IoT), and K be the number of active neighbors. Our approach to neighbor discovery is to judiciously design distinct codewords of length m bits for the nodes so that the receiver can identify the neighboring nodes from the received signal. active neighbors. Let $c_k = [c_k(i)]_{i=1}^m \in \{0,1\}^m$ be the codeword of user k. Let $S = \{i_1, i_2, ..., i_K\}$ be the set of active users. Suppose that each bit corresponds to a time sub-slot. An active node transmits its codeword in m sub-slots. The receiver performs a simple energy-based non-coherent detection at each sub-slot, and detects whether at least one user has transmitted signal (bit 1) in that sub-slot or not (bit 0). This corresponds to a bit-level OR channel as follows: the ith component of the received signal $y \in \{0,1\}^m$ is

$$y_i = \bigvee_{k \in S} c_k(i), \tag{1}$$

where $\vee$ is the boolean OR operator. The goal is to design the codewords $\{c_k\}_{k=1}^n$ such that the receiver is able to recover any K active neighbors using as few number of bits, m, as possible, and the receiver has low decoding complexity. It is clear that the above formulation is equivalent to the classical group testing problem, where y_i, $1 \le i \le m$ is the result of each test, and the K active neighbors correspond to the K defective items in group testing. Thus, by carefully designing the codewords of the n users, we determine which (active) users transmit an on signal (bit 1) in a sub-slot. This is equivalent to designing which items will be pooled in a single test of a group testing problem. See Figure 2 for an illustration.

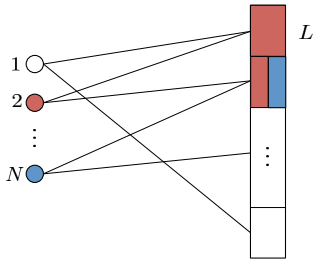


Fig. 2: **Schedule induced by random graph.** Each of the colored nodes represents an active node, whose signature is transmitted at the indicated slot on the right. The receiving node sees the concatenation of the bins on the right as a contiguous bit string.

B. Asynchronous Neighbor Discovery

The formulation is similar to what was explained before, with the difference that the users are not perfectly synchronized due to different path delays that makes synchronizing all the users a complex task. This is particularly the case in communication systems that are energy-limited such as the IoT. We assume that active user k's codeword is received with a delay of δ_k bits. We further assume that δ_k is uniformly distributed in the set $\{0, 1, \ldots, \Delta\}$, where Δ is the maximum delay offset between the users. Thus, the receiver detects a signal of $m + \Delta$ bits, where

$$y_i = \bigvee_{k \in S} c_k(i - \delta_k) \tag{2}$$

for $i = 1, \ldots, \Delta + m$, where $\Delta + m$ is the maximum transmission length. We note that $c_k(i) = 0$ if $i > m$. The goal is still to design the codewords c_k such that the receiver (access point) is able to recover any K active users using as few number of bits m as possible, and the receiver has low decoding complexity. Note that δ_is are not known to the access point, i.e., for decoding, the access point should not require the estimate of what the delay of each user is.

III. Overview of the SAFFRON Algorithm for Synchronous Neighbor Discovery

In this section, we give a brief overview of the SAFFRON algorithm for group testing. We first describe the noiseless algorithms and later robustify them in Section IV-A. The key idea of SAFFRON is to design the tests based on sparse-graph codes, and recover the active users using a simple peeling-based iterative algorithm.

We now describe how the tests in the SAFFRON scheme are designed, and how the decoder finds the active users. Consider a bipartite graph with n left nodes and M right nodes. Here, the n left nodes correspond to the n items, and the M right nodes corresponds to the M bundles of test results. We design a bipartite graph based on a random left-regular construction. That is, each left node is connected to constant number d of the right nodes uniformly at random, independently of the other nodes.

We denote the incidence matrix of a bipartite graph $\mathcal{G}$ by $T_{\mathcal{G}} \in \{0,1\}^{M \times n}$, or simply T if $\mathcal{G}$ is clear from the context. Let t_i be the ith row of $T_{\mathcal{G}}$. We associate each left node with a carefully designed signature (column) vector u of length h, i.e., $u \in \{0,1\}^h$. Note that the signature vector of a user (left node) is different from the codeword of the

user. Let us denote the signature vector of item i by u_i. We define the signature matrix

$$U \stackrel{\text{def}}{=} \begin{bmatrix} u_1 & u_2 & \dots & u_{n-1} & u_n \end{bmatrix} \in \{0,1\}^{h \times n}$$

Let the testing matrix $A \in \{0,1\}^{m \times n}$ be the matrix constructed by n column vectors that are codewords of the users. Given a graph $\mathcal{G}$ and a signature matrix U, we design the testing matrix A to be a row tensor product of $T_{\mathcal{G}}$ and U, which is defined as

$$A = T_{\mathcal{G}} \otimes U \stackrel{\text{def}}{=} \begin{bmatrix} A_1^T & A_2^T & \dots & A_M^T \end{bmatrix}^T \in \{0,1\}^{hM \times n}$$

where $A_i = U\text{diag}(t_i) \in \{0,1\}^{h \times n}$, and diag$(\cdot$ is the diagonal matrix constructed by the input vector. As an example, the row tensor product of matrices:

$$T = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } U = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \tag{3}$$

is

$$A = T \otimes U = \left[\begin{array}{ccc} 0 & \mathbf{0} & 0 \\ 0 & \mathbf{1} & 0 \\ \hline \mathbf{1} & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{1} & 0 \\ \hline 0 & 0 & \mathbf{1} \\ 0 & 0 & \mathbf{1} \end{array}\right] \tag{4}$$

For notational simplicity, we define the measurement vector corresponding to right node i as $z_i \stackrel{\text{def}}{=} y_{(i-1)h+1:ih}$ for $1 \leq i \leq M$. In other words, z_i is the bitwise logical ORing of all the signature vectors of the active users that are connected to right node i.

Our decoding algorithm simply iterates through all the right node measurement vectors $\{z_i\}_{i=1}^M$, and checks whether a right node is *resolvable* or not. A right node is resolvable if exactly one new (not previously detected) active user can be detected by processing the measurement of the right node. The decoding algorithm is terminated when there are no more resolvable right nodes.

To precisely explain our decoding algorithm, we present the following terminology. A right node that is connected to one and only one active left node is called a *singleton*. A right node that is connected to two active left nodes is called a *doubleton*. In the following, we explain how a singleton and its corresponding active user can be detected.
Consider the signature matrix where the ith column is a vertical concatenation of b_i and its complement $\bar{b}_i$, where b_i is the $L = \lceil \log_2 n \rceil$-bits binary representation of $i-1$, for $i \in [n]$.

We now show that a singleton can be detected and resolved with the aid of this signature matrix. First, note that the sum of the weight of any binary vector and the weight of its complement is always the length of the vector, L. Thus, given a singleton, the weight of the measurement vector is L. *Furthermore, if the right node is connected to zero or more than one active neighbor, the weight of the measurement vector is not L.* Therefore, by just checking the weight of the right-node measurement vector, one can simply detect whether the right node is a singleton or not. Further, one can also read the first half of the measurement vector of the detected singleton to find the corresponding active user. In [14], it is also shown that doubletons can also be detected, recovering a new active user. This can be achieved by expanding the signature matrix. We refer the readers to [14] for the details of how the signature matrix is expanded, and how the doubletons can be recovered. Our main result on the synchronous active user detection is the following theorem that is identical to Theorem 4.1. in [14].

Theorem 1. *With codewords of length $m = C(\epsilon)K \log_2 n$, the SAFFRON scheme recovers at least $(1-\epsilon)K$ active users with probability $1 - \mathcal{O}(\frac{K}{n^2})$, where ϵ is an arbitrarily-close-to-zero constant, and $C(\epsilon)$ is a constant that depends only on ϵ. Table II shows some pairs of ϵ and $C(\epsilon)$. The computational complexity of the decoding algorithm is linear in the length of the codewords, i.e., $\mathcal{O}(K \log(n))$, that is order-optimal.*

TABLE II: Pairs of ϵ and $C(\epsilon)$

Error floor, ϵ	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}
$C(\epsilon)$	36.78	47.28	57.78	68.16	78.60	89.04

IV. Asynchronous SAFFRON for Neighbor Discovery

In this section, we explain how one can adapt the SAFFRON scheme to address the case that the codewords of the active users are not synchronized. Recall that in the asynchronous case, the i-th bit that the decoder receives is $y_i = c_k(i - \delta_k)$. We maintain the $k \in S$ framework of designing the codewords of the users based on a sparse-graph codes construction. A key observation about the decoder of SAFFRON that helps us tackle the asynchronous case is as follows. For detecting whether a right node of the bipartite graph is a singleton or not, the decoder does not use any information about the structure of the graph. Instead, it only checks whether the weight of the $2L$-bits signature associated with the right node is exactly L. In the asynchronous case, although the codewords are still designed similarly based on a sparse-graph code, there is no clear notion of received bits corresponding to a right node of the graph, as codewords are shifted by arbitrary offsets.

We now explain the details of the asynchronous SAFFRON scheme. We first describe the Singleton-Only scheme, where the decoder only detects the singletons, and terminates decoding after all the singletons are detected. Without loss of generality suppose that $\Delta = 2k_0 \log(n)$ for some integer $k_0 > 1$. We design the bipartite graph (matrix T) of size $(k_0 + M) \times n$ as follows. Each entry of T is Bernoulli-distributed and i.i.d. with parameter p (to be determined), or equivalently, each possible edge of the bipartite graph is present with probability p independently. The signature vector of each user is the same as explained before, and has length $2\log(n)$.

Decoding Algorithm: We use a simple sliding-window decoder to detect the active users (as illustrated in Figure 3). The decoder checks all the $2L+7$ consecutive received bits. It first checks whether the first three bits and last three bits are 0, ensuring that it is an isolated singleton. It then checks whether the fourth bit is a 1, indicating the start of a signature. Finally, it looks at the weight of the remaining $2L$ bits. If the weight of the $2L$-bits binary string is L, the decoder declares a singleton and finds the corresponding active user by observing the first L bits that represent the binary expansion of the index of the active

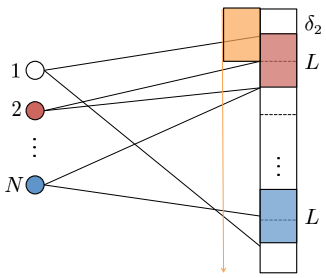


Fig. 3: **Singleton-only asynchronous graph.** Each of the colored nodes represents an active node, whose signature is transmitted at the indicated slot on the right. The receiving node sees the concatenation of the bins on the right as a contiguous bit string and decodes using a sliding window.

user. Note that the sliding window used by the decoder can mark the hypotheses while iterating through the received signal. It may then iterate through each of the hypotheses and decode.

For the purpose of analysis, we do not decode the first $2k_0 \log_2(n) = \Delta$ bits, and only decode the latter $2M \log_2(n)$ bits. Note that the total number of received bits can be as large as $2\Delta + 2M \log(n)$ due to asynchronization, which denotes the rate (overhead) required by SAFFRON to detect the active users. We also do not decode the final Δ bits.

We now state the main result of the paper:

Theorem 2. *With codewords of length $m = \Delta + \alpha K \log(K) \log_2(n)$, for $\alpha > 0$, the SAFFRON scheme recovers all of the K active users with probability $1 - \mathcal{O}(\frac{1}{K^{\Theta(1)}})$ while identifying no false positives. The computational complexity of the decoding algorithm is linear in the length of the codewords, i.e. it is $\Theta(\Delta + K \log(K) \log_2(n))$*

Proof: We first prove that the SAFFRON scheme recovers correctly recovers all of the K active users with probability $1 - \mathcal{O}(\frac{1}{K^{\Theta(1)}})$. Label the edges of the graph as $e_1, e_2, ..., e_{|E|}$, where E denotes the set of the edges of the graph. Let X_i be the random variable denoting the starting position (in bits) of the signature vector of e_i. Note that each edge of the graph corresponds to a signature vector of length $2\log_2(n)$ bits. Let $R = 2M \log_2(n)$ be the number of bits that we use for decoding. Let E_1 be the set of edges e_i such that $X_i \in [R]$. Consider a random edge e_ℓ in E_1. It is easy to see that with the described construction of the graph and with the assumption that δ_k is uniformly distributed in $\{0, 1, \ldots, \Delta\}$, X_ℓ is uniformly distributed in $[R]$. To analyze the performance of the decoding algorithm, we find the probability that a random edge e_ℓ corresponds to a singleton, i.e. there exists no X_i, $i \neq \ell$ such that $X_\ell - 2\log_2(n) \leq X_i \leq X_\ell + 2\log_2(n)$. That is, no other edges of the graph lie in a $4\log_2(n)$- neighborhood of X_ℓ. Fix another edge e_j. The probability that $X_\ell - 2\log_2(n) \leq X_j \leq X_\ell + 2\log_2(n)$ is $\frac{4\log_2(n)}{R}$. Thus, the probability that no X_i's lie in a $4\log_2(n)$-neighborhood of X_ℓ is:

$$\left(1 - \frac{4\log_2(n)}{R}\right)^{|E_1|-1} \simeq \exp\left(-\frac{4\log(n)|E_1|}{2M\log(n)}\right) \quad (5)$$

as n approaches infinity. We denote this quantity by $\bar{q}$ and note that we design $\bar{q}$ to be $\Theta(1)$. Note that this is guaranteed by designing $p = \Theta\left(\frac{1}{K}\right)$ so that $|E_1| = \Theta(M)$. Further, we may design $M = \alpha K \log(K)$. for some $\alpha > 0$ to be determined. Thus:

$$|E_1| = \beta K \log(K) = K\bar{d},$$

where the average degree of the active left nodes is $\bar{d} = \Theta(\log(K))$. Now, one can design the tuple (α, β) rather than (α, p); then, $\bar{q} = \exp(-\frac{2\beta}{\alpha})$. Now, conditioning on the degree of an active left node of the graph being x, the probability that the corresponding active user is not detected is $(1-\bar{q})^x = q^x$. As K tends to infinity, the degree of the active left nodes of the graph is Poisson distributed with parameter $\bar{d}$. Letting A_i be the event that left node i is not recovered after decoding and noticing that $\Pr(A_1) = \Pr(A_i)$ for all i, we see that:

$$\Pr(A_1) = \sum_{x=0}^{\infty} q^x e^{-\bar{d}} \frac{\bar{d}^x}{x!} \quad (6)$$

$$= e^{-\bar{d}\bar{q}} \quad (7)$$

$$= \exp\left(-e^{-\frac{2\beta}{\alpha}} \beta \log(K)\right) \quad (8)$$

Thus, using the union bound, the probability that an active left node is not detected can be upper bounded as follows:

$$\Pr\left(\bigcup_{i=1}^{K} A_i\right) \leq \sum_{i=1}^{K} \Pr(A_i) \quad (9)$$

$$= K \exp\left(-e^{-\frac{2\beta}{\alpha}} \beta \log(K)\right) \quad (10)$$

$$= K^{1-\beta e^{-\frac{2\beta}{\alpha}}} \quad (11)$$

$$\leq \frac{1}{K^{\Theta(1)}} \quad (12)$$

Where the last inequality follows by designing sufficiently large constants α and β.

We now show that the SAFFRON scheme does not decode any false positives. Let us denote the $2L + 7$ bits section covered by the sliding window as c'. Recall that the sliding window decoder checks whether the first and last three bits of c' are 0. If they are all 0 and the fourth bit is 1, then the decoder checks whether the remaining $2L$ bits of c' have weight L, in which case a singleton is detected. If the weight is not L, then there are at least two codewords that are synchronized and conflicting, and we do not decode. Otherwise, there is at least one codeword conflicting with c', and we do not decode. Thus, the only case where the decoder will attempt to decode c' is when the fourth bit is 1, the zero-pad is met, and the weight of the remaining $2L$ bits is L, which by construction, can only occur if c' is a singleton. □

We are ultimately interested in the minimum feasible α, so the following lemma helps us to establish this:

Lemma 3. *For any values of n and K, the minimum feasible $\alpha = 2e$*

Proof: We may write the following optimization problem:

$$\min \alpha \quad (13)$$

$$\text{s.t. } \alpha \geq 0, \beta \geq 0$$

$$\beta e^{-\frac{2\beta}{\alpha}} \geq 1$$

Now, note that $\max_x xe^{-x} = \frac{1}{e}$. Thus, we can see that:

$$\max \beta e^{-\frac{2\beta}{\alpha}} = \frac{\alpha}{2e}$$

Now, in order to minimize α while satisfying the constraints of the optimization problem, we may set $\alpha = 2e$. Thus, we can see that there exists $\beta > 0$ where the maximum value of $\beta e^{-\frac{2\beta}{\alpha}}$ can be achievably set to 1. □

A. Robustified Singleton-Only Asynchronous SAFFRON

In this section, we consider robustifying the singleton-only solution to noise. Notice that since the receiver is performing simple energy detection, we may model the various sources of noise (i.e. thermal noise, range, etc.) as i.i.d. bit flips with probability a at the receiver's end. In order to tolerate the noise, we add 2 random checks at the end of each codeword, and use an inner code on each codeword. More specifically, we utilize the following properties of spatially coupled codes [15] (with message length N and rate R):

- It has an encoding function $f(\cdot) : \{0,1\}^N \to \{0,1\}^{N/R}$ and a decoding function $g(\cdot) : \{0,1\}^{N/R} \to \{0,1\}^N$, and its decoding complexity is $O(N)$.
- If R satisfies:

  $$R < 1 - H(a) - \delta = 1 + a\log_2 a + (1-a)\log_2(1-a) - \delta$$

  for an arbitrarily small constant $\delta > 0$, then $\Pr(g(\mathbf{x} + \mathbf{w}) \neq \mathbf{x}) < 2^{-\zeta N}$ as $N \to \infty$, for some constant $\zeta > 0$.

We can now describe the augmented signature matrix for use in the noisy regime. Consider some arbitrary active node with index ℓ. Now, draw two random variables i_ℓ, j_ℓ independently from $[N]$. At a high level, we will use these as "random checks" to verify whether our decoded singleton is correct. Additionally, we append a set of all 1's of size $t \log n$ for some appropriately determined constant t to the beginning of the codeword. We finally use our inner code f on each of these checks so we have the following signature matrix:

$$U' = \begin{bmatrix} \frac{f(\mathbf{b}_1)}{f(\mathbf{b}_1)} & \frac{f(\mathbf{b}_2)}{f(\mathbf{b}_2)} & \cdots & \frac{f(\mathbf{b}_{n-2})}{f(\mathbf{b}_{n-2})} & \frac{f(\mathbf{b}_{n-1})}{f(\mathbf{b}_{n-1})} \\ \frac{f(\mathbf{b}_{i_1})}{f(\mathbf{b}_{i_1})} & \frac{f(\mathbf{b}_{i_2})}{f(\mathbf{b}_{i_2})} & \cdots & \frac{f(\mathbf{b}_{i_{n-2}})}{f(\mathbf{b}_{i_{n-2}})} & \frac{f(\mathbf{b}_{i_{n-1}})}{f(\mathbf{b}_{i_{n-1}})} \\ \frac{f(\mathbf{b}_{j_1})}{f(\mathbf{b}_{j_1})} & \frac{f(\mathbf{b}_{j_2})}{f(\mathbf{b}_{j_2})} & \cdots & \frac{f(\mathbf{b}_{j_{n-2}})}{f(\mathbf{b}_{j_{n-2}})} & \frac{f(\mathbf{b}_{j_{n-1}})}{f(\mathbf{b}_{j_{n-1}})} \end{bmatrix}$$

Decoding Algorithm: Here, we briefly describe the augmented decoding scheme in robustified singleton-only SAFFRON. The decoder slides a window of length $\left(t + \frac{6}{R}\right)\log n$ and checks the energy of the first $t \log n$. If this energy is greater than an energy threshold $T_e = \frac{t \log n}{2}$, the following $\frac{6 \log n}{R}$ bits are added as a hypothesis, and the window keeps sliding. We now note that in order to ensure the singleton is isolated, the decoder skips the following $\frac{6 \log n}{R}$ bits, only adding them to the hypothesis list. It keeps track of how many bits are left to "restart", and resets this counter once it sees a set of $t \log n$ bits which pass the energy test. The decoder only marks a hypothesis when the "restart" counter hits 0 and it sees a set of contiguous $t \log n$ bits that pass the energy test. Now, note that since we are still using the complement of each codeword, a valid singleton will still have approximately constant weight. This allows the decoder to iterate through the decoding map and check for an approximately constant weight of $3\left(\frac{\log n}{R}\right)$. In this way, just as in the noiseless decoder, it may take one pass through the received bit string, and then carefully check the remaining hypotheses in the following manner. We first use the decoding map $g(\cdot)$ to find the hypothetical active node corresponding to the first $\left(\frac{\log n}{R}\right)$ bits. Then, using the two random checks in the following two $\left(\frac{\log n}{R}\right)$ length sections, the decoder checks the initial hypothesis. If the checks were inconsistent, it does not declare a resolved singleton, and continues decoding. Noting that the extension to asynchrony allows us to use the same designs to robustify as in the synchronous case, we state the following lemma and theorem summarizing this. The key component is Lemma 4, and its proof can be found in [14].

Lemma 4. *Robustified-SAFFRON misses a singleton with probability no greater than $\frac{3}{n^\zeta}$. Robustified-SAFFRON wrongly declares an active neighbor with probability no greater than $\frac{1}{n^{2+\zeta}}$.*

Theorem 5. *The sample and decoding complexity of Robustified, Asynchronous Singleton-only SAFFRON are both $\mathcal{O}(K \log(K) \log(n))$ where the exact constants can be specified as a function of α and a.*

We note that one can easily bound the number of hypothesis by $\mathcal{O}(K \log K)$ using Hoeffding's bounds. Hence, Theorem 5 is a direct application of Lemma 4 along with this. We omit the details.

V. Simulation Results

In this section, we empirically evaluate the asynchronous SAFFRON algorithm through extensive simulations. Simulations in this section are done in Python.

A. Noiseless Simulations

As noted in Section IV, the asynchronous SAFFRON algorithm has the following properties:

1) Decoding complexity: $\Theta(\Delta + K \log(K) \log_2(n))$.
2) Performance as a function of an oversampling parameter α, which can be explicitly determined.

We illustrate these key properties in the simulations. First, we fix the size of the problem with $n = 2^{32}$ and $K = 2^{10}$ and vary the parameter α. Recall that in the proof of 2, we mention the tuple (α, β). The optimal $\alpha = 2e$ can be clearly seen from the plot in Figure 4. Additionally, in order to illustrate the decoding complexity of the algorithm, we plot the runtime of the algorithm according to varying values of n and K. We note that as n varies, it clearly follows a log-linear trajectory, as shown in Figure 5b. As K varies, we fit a curve following the trajectory of $K \log K$, which the empirical data points follow tightly, as one can see in Figure 5a.

B. Noisy Simulations

We now show the performance of the Asynchronous SAFFRON algorithm in the presence of noise. Whereas in Section IV-A, we use a capacity achieving code to achieve linear encoding and decoding complexity, for the purposes of simulation, we use an inner Reed-Solomon code before specifying the code induced by the sparse graph. The simulations here are run with the total number of nodes

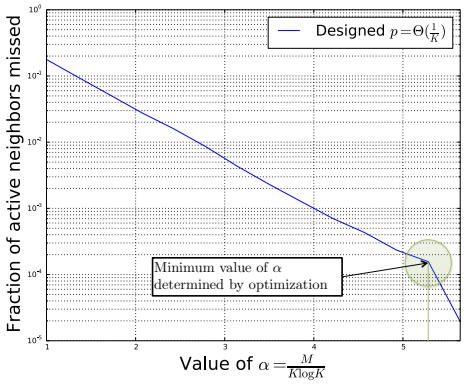
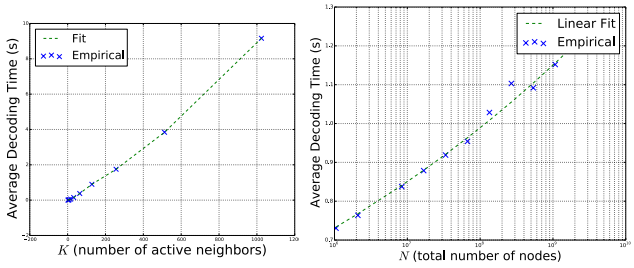


Fig. 4: **Fraction of Undetected Neighbors as a Function of Oversampling.** Using values of $n = 2^{32}$, and $K = 2^{10}$, and $\Delta = 200$, as well as designing $p = \Theta(\frac{1}{K})$, the fraction of undetected neighbors as a function of the oversampling ratio. Note the threshold, which can be determined by the optimization routine given in Equation 13.



(a) Noiseless, $n = 2^{32}$ (b) Noiseless, $K = 2^8$

Fig. 5: **Noiseless simulation results of Asynchronous SAFFRON.** (a), (b): We measure run-time of Asynchronous SAFFRON for varying values for n and K.

in the networks $n = 2^{24}$ and number of active neighbors $K = 2^5$. The utilized Reed-Solomon code uses a field size of c_q, and takes a message of size c_k encoding it into a codeword of size c_n, where $c_n \leq c_q$. For our simulations, we use $c_q = 2^8$, and view $\lceil \log_2 n \rceil$ as a 3 symbol message. We thus use $c_k = 3$ and plot the performance of the robust algorithm as c_n varies, noting that the error correcting capability of the code is given by $\lfloor \frac{c_n - c_k}{2} \rfloor$. The results can be seen in Figure 6.

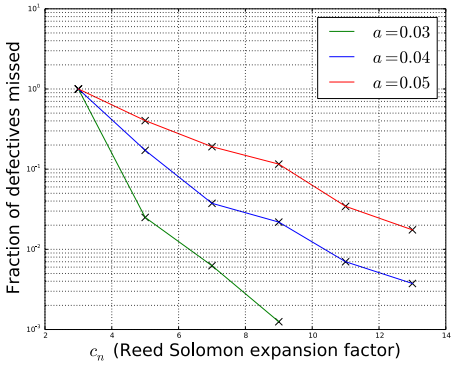


Fig. 6: **Fraction of Undetected Neighbors as a Function of c_n.** Using values of $n = 2^{24}$, and $K = 2^5$, and $\Delta = 200$, as well as designing $p = \Theta(\frac{1}{K})$, the fraction of undetected neighbors as a function of the expansion in the RS code c_n. Each curve corresponds to a different level of noise in the system.

VI. Conclusion and Further Work

In this paper, we presented a novel formulation of the neighbor discovery problem for an IoT setting, where the nodes operate in the low energy regime and are not fully synchronized. We made essential use of the recently-proposed SAFFRON scheme for group testing to design a new neighbor discovery protocol. Our asynchronous protocol has a transmission length of $\mathcal{O}(\Delta + K \log K \log n)$ and decoding complexity linear in the transmission length, where we are able to furnish exact constants for the oversampling ratio α. Additionally, we robustify our protocol to noise, showing that the transmission length and decoding complexity stay order-wise equivalent. Finally, we corroborate the theoretical results with extensive simulations.

Future work involves extending the idea outlined here with synchronization at the bit level to incorporate a peeling structure, augmenting the encoder and decoder to be able to handle collisions of at most two transmissions, and robustifying this protocol to noise.

References

[1] A. P. Athreya and P. Tague, "Network self-organization in the internet of things," in *2013 IEEE International Conference on Sensing, Communications and Networking (SECON)*, 2013.

[2] M. Baker, S. Sesia, and I. Toufik, "Lte-the umts long term evolution from theory to practice," 2011.

[3] Z. Zou, D. S. Mendoza, P. Wang, Q. Zhou, J. Mao, F. Jonsson, H. Tenhunen, and L.-R. Zheng, "A low-power and flexible energy detection ir-uwb receiver for rfid and wireless sensor networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 7, pp. 1470–1482, 2011.

[4] J. Wolf, "Born again group testing: Multiaccess communications," *IEEE Transactions on Information Theory*, vol. 31, no. 2, pp. 185–191, 1985.

[5] J. Luo and D. Guo, "Neighbor discovery in wireless ad hoc networks based on group testing," in *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, 2008.

[6] M. Tabesh, N. Dolatsha, A. Arbabian, and A. M. Niknejad, "A power-harvesting pad-less millimeter-sized radio," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 4, pp. 962–977, 2015.

[7] P. Dutta and D. Culler, "Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*, 2008.

[8] A. Kandhalu, K. Lakshmanan, and R. R. Rajkumar, "U-connect: a low-latency energy-efficient asynchronous neighbor discovery protocol," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2010.

[9] S. Vasudevan, M. Adler, D. Goeckel, and D. Towsley, "Efficient algorithms for neighbor discovery in wireless networks," *IEEE/ACM Transactions on Networking*, vol. 21, no. 1, pp. 69–83, 2013.

[10] S. A. Borbash, A. Ephremides, and M. J. McGlynn, "An asynchronous neighbor discovery algorithm for wireless sensor networks," *Ad Hoc Networks*, vol. 5, no. 7, pp. 998–1016, 2007.

[11] L. Zhang, J. Luo, and D. Guo, "Neighbor discovery for wireless networks via compressed sensing," *Performance Evaluation*, vol. 70, no. 7, pp. 457–471, 2013.

[12] A. Stajkic, F. Clazzer, and G. Liva, "Neighbor discovery in wireless networks: A graph-based analysis and optimization," in *Communications Workshops (ICC), 2016 IEEE International Conference on*, 2016.

[13] K. R. Narayanan and H. D. Pfister, "Iterative collision resolution for slotted aloha: An optimal uncoordinated transmission policy," in *Turbo Codes and Iterative Information Processing (ISTC), 2012 7th International Symposium on*, 2012.

[14] K. Lee, R. Pedarsani, and K. Ramchandran, "SAFFRON: A fast, efficient, and robust framework for group testing based on sparse-graph codes," *CoRR*, vol. abs/1508.04485, 2015.

[15] S. Kudekar, T. J. Richardson, and R. L. Urbanke, "Threshold saturation via spatial coupling: Why convolutional ldpc ensembles perform so well over the bec," *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 803–834, 2011.