FlipTracer: Practical Parallel Decoding for Backscatter Communication

Meng Jin, *Member, IEEE, ACM*, Yuan He, *Senior Member, IEEE, Member, ACM*, Xin Meng, Yilun Zheng, Dingyi Fang, *Member, IEEE, ACM*, and Xiaojiang Chen, *Member, IEEE, ACM*

***Abstract*— With parallel decoding for backscatter communication, tags are allowed to transmit concurrently and more efficiently. Existing parallel decoding mechanisms, however, assume that signals of the tags are highly stable and, hence, may not perform optimally in the naturally dynamic backscatter systems. This paper introduces FlipTracer, a practical system that achieves highly reliable parallel decoding even in hostile channel conditions. FlipTracer is designed with a key insight; although the collided signal is time-varying and irregular, transitions between signals' combined states follow highly stable probabilities, which offers important clues for identifying the collided signals and provides us with an opportunity to decode the collided signals without relying on stable signals. Motivated by this observation, we propose a graphical model, called one-flip-graph (OFG), to capture the transition pattern of collided signals, and design a reliable approach to construct the OFG in a manner robust to the diversity in backscatter systems. Then, FlipTracer can resolve the collided signals by tracking the OFG. We have implemented FlipTracer and evaluated its performance with extensive experiments across a wide variety of scenarios. Our experimental results have shown that FlipTracer achieves a maximum aggregated throughput that approaches 2 Mb/s, which is 6× higher than the state of the art.**

***Index Terms*— Backscatter, wireless, parallel transmission.**

I. Introduction

WITH the proliferation of the Internet of Things, we envision explosive growth of backscatter applications in terms of both the quantity of deployed tags and the amount of data volumes carried by backscatter. Due to the dense deployment of tags, many tags will co-existence within the coverage of the reader. In the future, more and more applications, such as warehouse inventory, object tracking, and smart surveillance in industry, will require high speed data offload from a variety of data-rich sensors [1]–[6]. Therefore, how to fully utilize the channels and enhance network throughput, turns to be a crucial problem in the relevant area.

One potential solution to provide high throughput is to parallelize multiple backscatter transmissions. This however leads to an important challenge in parallel transmission: how collided signals can be decoded. Existing work proposed to use coding techniques, but it incurs computational overhead on the tags, which may not be realistic. More recent works [7]–[9] propose to shift the workload from tags to backscatter readers. Specifically, the reader can *separate* the interleaved signal edges from different tags using its capability to sample at a much higher rate than a tag. Then, according to the signals' locations in the In-phase and Quadrature (IQ) domain, the reader can *decode* the transmitted signals of tags.

Unfortunately, in order to distinguish signals from different tags, the above parallel decoding proposals rely on *stable* and *distinct* features of signals in the time and/or the IQ domain. On the contrary, in our real-world experiments, we have observed that signal features are likely to be highly *dynamic* and *unpredictable*. As a result, applying the existing proposals in practice usually results in unacceptably high decoding error rates, which in turn affects transmission efficiency and network throughput negatively.

It is therefore highly desirable to design practical protocols for parallel backscatter transmissions with hostile channel conditions. In our experiments, we have made a surprising observation. Despite the lack of stable features in the time and IQ domains, transitions between the *combined states* of these signals follow highly *stable* probabilities. To be more specific, due to the intrinsic asynchronism of signals from different tags, a higher transition probability between two combined states indicates higher similarity between their corresponding signals. Tracking these transition probabilities may help to distinguish signals' states, offering plenty of information for parallel decoding.

Towards the design of a practically usable solution, we still need to address the following challenges. First, factors like noise and frequency drifts of antennas may negatively affect signals' states, which need to be identified in a consistently correct fashion. Second, it is challenging to capture the stable transition probabilities in unstable signals, and to design a new model to represent these probabilities. Last but not the least, signal processing and counting are often error-prone, which must be taken into account when designing a practical decoding scheme.

To address the above challenges, we in this paper propose FlipTracer, a practical system that uses the observed *transition probabilities* between signals' combined states as input, and based on this information alone, achieves highly reliable parallel decoding. To realize this vision, we construct a graphical model, called *one-flip-graph (OFG)*, to capture the signals'

Manuscript received December 25, 2017; revised July 7, 2018 and November 27, 2018; accepted December 6, 2018; approved by IEEE/ACM Transactions on Networking Editor B. Shrader. Date of publication January 11, 2019; date of current version February 14, 2019. This work was supported in part by the National Key R&D Program of China under Grant 2017YFB1003000, in part by the National Natural Science Fund of China for Excellent Young Scientist under Grant 61422207, in part by the NSFC under Grant 61772306, Grant 61572402, Grant 61672428, and Grant 61672427, and in part by the Program for Innovative Research Team in University under Grant IRT13090. *(Corresponding authors: Yuan He; Dingyi Fang.)*

M. Jin and Y. He are with the School of Software and BNRist, Tsinghua University, Beijing 100084, China (e-mail: heyuan@tsinghua.edu.cn).

X. Meng is with the School of Information Science and Technology, Northwest University, Xi'an 710069, China.

Y. Zheng is with the School of Software and BNRist, Tsinghua University, Beijing 100084, China.

D. Fang and X. Chen are with the School of Information Science and Technology, Northwest University, Xi'an 710069, China (e-mail: dyf@nwu.edu.cn).

Digital Object Identifier 10.1109/TNET.2018.2890109

transition pattern across the combined states, and provides fine-grained information about the similarity between signals' combined states. Based on the one-flip-graph, FliperTracer can decode the collided signal without relying on the stability of signal features, and is highly efficient and robust in a wide variety of practical scenarios. To the best of our knowledge, FlipTracer is the first practical solution that achieves highly reliable parallel decoding in practical scenarios.

In summary, our key contributions are two-fold:

- With a new graphical model (OFG) that translates the transition probability between combined states to the similarity between their corresponding signals, we are able to guarantee high efficiency under parallel decoding in hostile channel conditions.
- In practice, we have designed and implemented FlipTracer, a practical system that integrates two main techniques: reliable OFG construction based on the trajectory of the signals in the IQ domain, and collided signal identification by tracking the constructed OFG. Our experimental results have shown that its achievable aggregate throughput approaches 2Mbps, which is 6× higher than the state-of-the-art [7].

In the rest of the paper, we discuss related work in Section II, and describe the motivation of our design in Section III. The design details of FlipTracer are introduced in Sections V∼VII. Sections IX and X evaluate the performance analytically and empirically. We conclude the paper in Section XI.

II. Related Work

There have been many efforts made to enable concurrent transmission in backscatter communication.

A. Using Coding Mechanisms

A direct method for parallel transmission is to exploit coding mechanisms [10]–[13] on tags to facilitate collision recovery. In these methods, each tag exploits an orthogonal code for encoding data where each bit is spread using a long PN sequence. Although such methods enable concurrent transmission, they incur unaffordable overhead on the tag side.

To solve this problem, some recent works propose to decode concurrent transmissions by extending the decoding capacity at the reader-side [7]–[9], [14]–[18]. Specifically, in these works, the reader decode the collided signals according to the signals' features in time and/or IQ domain:

B. Using Tags' IQ Domain Features

Theoretically the channel coefficients of the colliding tags are *stable* and collided signal is the *linear combination* of these tags' channel coefficients. Thus the collided signals with specific combined states will exhibit specific positions on the IQ plane. Under this assumption, designs like [9], [14], [15] propose to identify the collided signals based on their IQ domain positions.

C. Using Tags' Time Domain Features

An alternative method that has been considered in prior works is separation in the time domain [7], [9]. They assume that the signal edges of the same tag will come at a relatively fixed interval (i.e., *stable bit duration*). Thus they can separate the signal edges of different tags in the time domain,

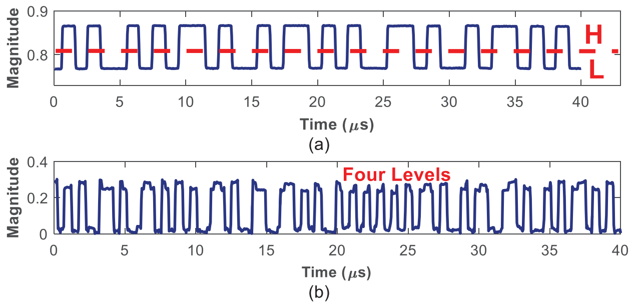


Fig. 1. Signals in time domain: (a) a single tag. (b) two tags.

and connect with the signal clusters in the IQ domain for collision decoding. For example, LF-Backscatter [9] applies folding to extract the periodical signal edges of individual tags. BiGroup [7] achieves this using the linear regression model.

Although the above methods enable parallel decoding, our study reveals that they cannot provide satisfactory performance in practical backscatter systems. The reason is that they have to rely on the stability of the tags' signals, which is however unrealistic in practice. Hence, applying the above methods in practice generally results in high decoding error rates, which in turn hurts the transmission efficiency and throughput. Different from all the above works, FlipTracer neither requires modifications on the tag side, nor assumes stable features of the collided signals. FlipTracer exploits the transition pattern among the signals which is fairly stable and robust against the dynamics in both the time and the IQ domains. Our experimental results show that FlipTracer is able to achieve highly reliable parallel decoding under various conditions.

III. Background and Motivation

A. Preliminary

In backscatter systems, the tag encodes its data by reflecting or absorbing the carrier waves, resulting in two possible states: "High (H)" and "Low (L)", which can be decoded using a magnitude threshold (as shown in Figure 1(a)). When N tags transmit concurrently, their signals add up at the reader, and the collided signal will have 2^N combined states, forming 2^N signal levels. One combined state is theoretically a *linear* combination of all the N tags' signal state, denoted as $\mathbf{S} = [S_1, S_2, \ldots, S_N]$, where $S_i = H_i$ or L_i indicates the state of tag i. Figure 1(b) shows the example of a 2-tag collision case. We find that we cannot determine the threshold to detect the states of either individual tag.

We further show the IQ signals received from 2 tags in Figure 2(a). We find that the symbols form four separable clusters, each represents a combined state of the tags. Thus if we can identify the combined state of each cluster, we can know whether an individual tag i's transmission state $S_i = H_i$ or L_i for each $i = 1, \ldots, N$. Then the collision is decoded. This is however a challenging task due to the dynamics in the backscatter signals.

B. Signal Dynamics

The challenges for collision decoding in practical backscatter systems mainly lie in the highly dynamic signal of tags, in both the IQ and the time domains.

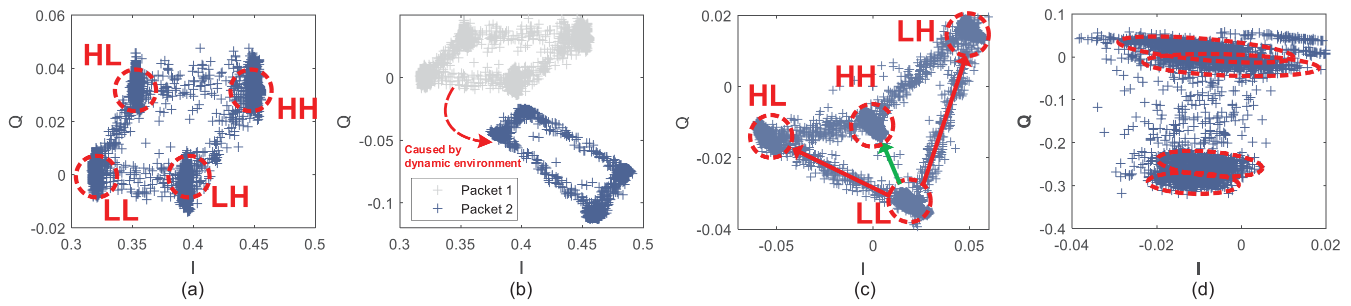


Fig. 2. Symbol clusters: (a) a 2-tag collision example. (b) collided signal from two sequential packets under dynamic working condition. (c) 2-tag collision with non-linear dependency. (d) overlapped clusters.

1) Dynamics in the IQ Domain: Theoretically, the combined states of the clusters can be identified based on their locations in the IQ domain if the channel coefficients of tags are stable and are linearly combined when collision occurs. However, we observe both *dynamic* and *non-linear depended* signals in the IQ domain:

1) The positions of the clusters in the IQ plane will change with the dynamic environment. Figure2(b) shows the IQ symbols collected from two sequential packets (0.2s apart) when an obstacle moves around the reader. We can observe great difference between the positions of the two packets' symbol clusters. In this case, the decoding methods [14] which rely on channel coefficients have to conduct channel measurements frequently to deal with the changing environment. This significantly increases the protocol overhead.

2) The locations of clusters may sometimes exhibit non-linear dependence. In practice, when multiple tags coexist, tags may backscatter the signals from nearby tags, resulting in the non-linear dependency in the combined signals received at the reader [7]. Figure 2(c) gives an 2-tag collision case. In such a scenario, the reader cannot identify the clusters based on their position on the IQ plane.

3) The clusters may be too close to be separated. When the number of tags increases, the number of clusters will increase exponentially, resulting in clusters being closer to (or even overlap with) each other. What is worse, in a low SNR scenario, noise from the environment will increase the radius of the clusters, which incurs overlap even in 2-tag collision scenarios (as shown in Figure 2(d)). In such scenarios, we can not even separate different clusters.

2) Dynamics in the Time Domain: Distinguishing the collided signals in time domain is also challenging. Specifically, to map each signal edge to the flipping of each tag, an important assumption is that the tags have relatively stable bit durations. This requires the built-in oscillators have low drifting rate (less than 200 ppm is tolerable [9]). However, due to the low cost design, the oscillators used in backscatter tags typically exhibit high drifting rates [9], [19], [20]. Figure 3 shows the CDF of the measured drifting rates of 20 tags. We can observe that 70% of tags' clock drifting rates fall between 40,000 ppm and 68,000 ppm. Consider a concrete example where tags transmit 100-bit packets at 100Kbps. Such high drifting rates can lead to $40 \sim 68\mu s$ time offsets within one-packet duration. Since the bit duration is only $10\mu s$, $40 \sim 68\mu s$ offsets will inevitably incur excessive errors in the edge identification process.

Some schemes [7] propose to address this problem by using linear regression. Such schemes work only if the signal

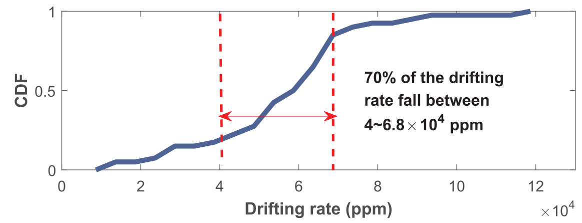


Fig. 3. CDF of the measured drifting rate of 20 tags.

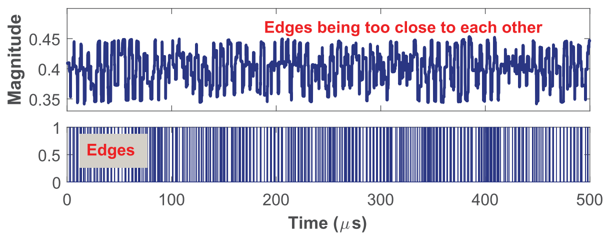


Fig. 4. Signal series of a 4-tag collision.

flips infrequently. Under this assumption, the long distance between edges can be used to gain confidence in edge detection. However, when multiple tags transmit concurrently at high bitrates (as shown in Figure 4), signal edges are stacked side by side. In this situation, the reader can hardly distinguish the edges from different tags.

C. Transition Probabilities Between Clusters

The above experimental results present a conundrum that: how to identify the combined states of each cluster based on such dynamic and irregular signals? Our idea is to leverage signals' *transition probabilities* between clusters. Specifically, Figure 5 shows the transfer trajectories of collided signals from two tags under different scenarios. We find that although the signals are dynamic and irregular, signals' transition probabilities between different clusters are highly *stable* and *traceable*. Indeed, a higher transition probability between two clusters implies the higher similarity between their corresponding combined states.

Let us explain the reason behind. In practice, flipping of tags makes the collided signal to transfer among clusters. Specifically, the flipping of a single tag causes the transition between two clusters whose combined states have only one different state (e.g., “LL” and “LH”), while the concurrent flipping of multiple tags causes the transition between two clusters whose combined states have multiple different states (e.g., “LL” and “HH”). We term the former cluster pair

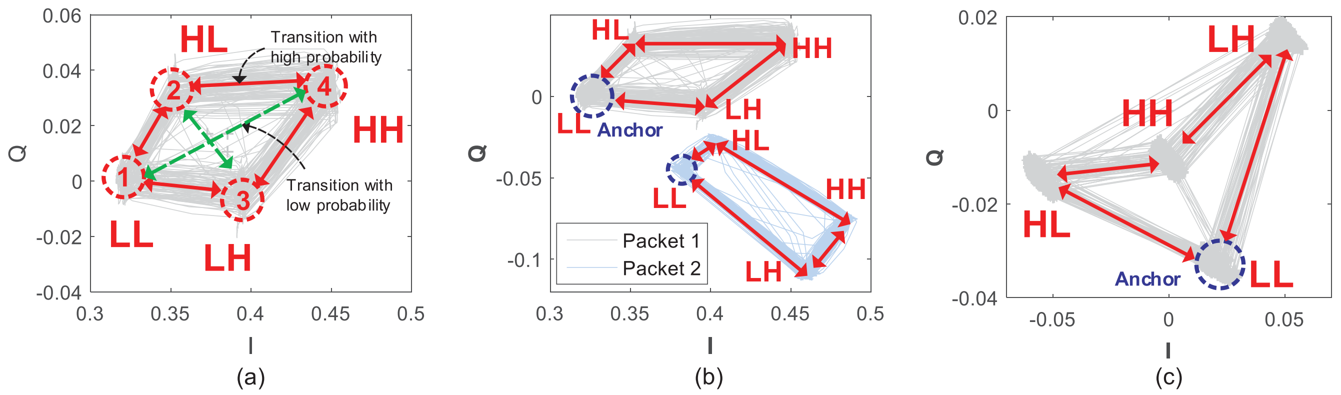


Fig. 5. Transition pattern among clusters: (a) a 2-tag collision example. (b) collision under a dynamic working condition. (c) collision with non-linear dependency

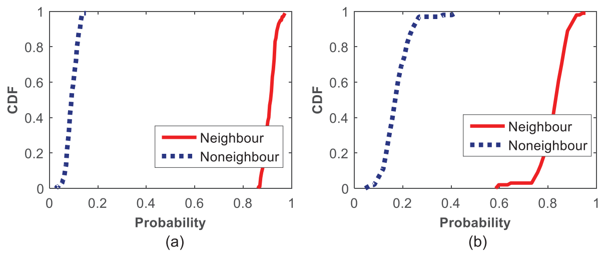


Fig. 6. The CDF for the transition probability between both neighbor and non-neighbor clusters: (a) two tags; (b) five tags.

as *neighbor clusters* and the latter as *non-neighbor clusters*. Given that the state flipping of different tags are interleaved most of the time [7], [9], the transition probabilities between the neighbor clusters are obviously higher than those between the non-neighbor clusters.

As a concrete example, consider a 2-tag collision case where each tag transmits at 100Kbps and the USRP reader samples at 25 MHz. An edge has a width of 3 samples at the reader's sampling rate, which means that we can stack at most $\frac{250}{3} = 83$ separated edges in one bit duration. Thus the probability for the two tags to flip their state at exactly the same time can be calculated as: $1-(83\cdot 82)/(83\cdot 83) = 1.2\%$. Moreover, detailed discussion is given in Section IX-C.

Figure 6(a) plots the CDFs of the transition probabilities between both neighbor and non-neighbor clusters, which is calculated based on more than 100 collided packets of two concurrent tags (bitrate of the tags is 100Kbps). We can see that the transition probability between neighbor clusters is almost $8\times$ higher than that between non-neighbor clusters. Figure 6(b) plots the CDFs of the transition probabilities in a 5-tag collision case. We can see that the above phenomenon still exists even when the concurrency level is high.

IV. FlipTracer Overview

FlipTracer is a practical parallel decoding approach in which the collided signals are identified based on the transition probabilities between symbol clusters. The FlipTracer system architecture is shown in Figure 7:

- **OFG construction.** FlipTracer first clusters the received samples. Then, based on the observed signal transition probabilities among signal clusters, FlipTracer constructs a graph, termed as *one flip graph (OFG)*, which connects all the neighbor clusters. The OFG acts as an important reference for cluster identification.

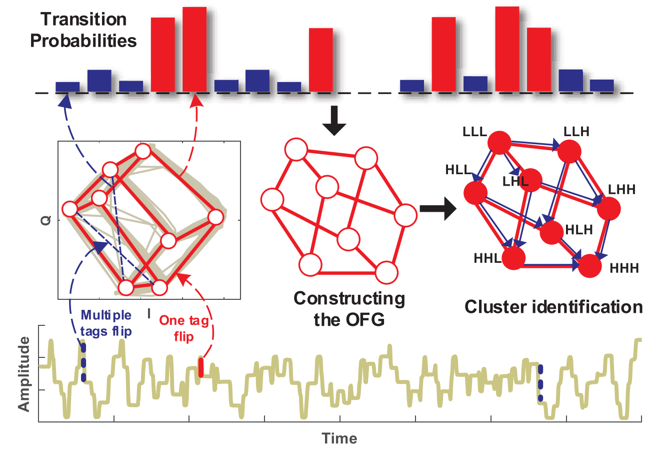


Fig. 7. Workflow of FlipTracer.

- **Cluster identification.** By tracking the OFG, FlipTracer identifies the combined states of all the clusters in the OFG. Then, by examining the identified combined states, FlipTracer outputs N sequences of binary states that represent the transmitted signals of the N colliding tags.
- **Decoding.** For each tag, FlipTracer applies a Conditional Random Field based decoder on the binary sequence to identify the most likely sequence of bits.

The next few sections elaborate on the above components, providing the technical details.

V. The One-Flip-Graph

The One-Flip-Graph (OFG) is denoted as $\mathbf{OFG} = (\mathbf{C}, \mathbf{E})$, where $\mathbf{C} = \{C_1, \ldots, C_{Nc}\}$ denotes the $Nc = 2^N$ symbol clusters and if C_i and C_j $(C_i, C_j \in \mathbf{C})$ are neighbor clusters, we have $< C_i, C_j > \in \mathbf{E}$. In this section, we describe how to construct the OFG based on the collected IQ symbols.

A. Symbol Clustering

The first challenge we face is reliable symbol clustering, which is central for constructing correct OFG. Although symbol clustering methods have been proposed in some recent works [7], [9], they can not separate overlapped clusters and therefore results in poor performance under noisy environments. In addition, the performance of the existing methods depends on the selected parameters. However, it is generally difficult to have an appropriate setting of the parameters.

We propose a new clustering method to solve the above problems.

1) Center Identification: We start by identifying the cluster centers based on the algorithm introduced in [21] (we call it LDBC), which is able to identify centers of overlapped clusters. In LDBC, cluster centers are characterized by a higher density than their neighbors and a large distance from samples with higher density. Specifically, for each signal sample s_i, we compute two quantities: its local density ρ_i, and its distance δ_i to the samples with higher density, where ρ_i is defined as the number of samples that are closer than d_c (a cutoff distance)[1] to sample s_i, and δ_i is the minimum distance between sample s_i and any other samples with higher density. δ_i is much larger than the typical nearest neighbor distance only for samples that are local or global maxima in terms of density.

Now, it seems that we can identify the cluster centers as the samples which have both high δ_i and high ρ_i. However, a problem we face here is, when multiple clusters overlapped with each other, the overlapping area also has high density since it gathers the outlier samples from multiple clusters. This makes it difficult to correctly identify the centers. To solve this problem, we further exploit time domain features for cluster center identification. Specifically, in a signal cluster, the movement of the signal follows Gaussian process. That is to say, the signal will stay in the central area, but occasionally moves to an outlier location (maybe in an overlapping area). Therefore, the samples that are located at the central area will exhibit higher continuity compared with those are located at the overlapping area, although these two areas show similar sample density.

We propose a metric (termed as Con_i) to describe the continuity of a sample i as follows:

$$Con_i = \frac{\sum_{j=i-w/2}^{i+w/2} \chi(d_{ij} - d_c)}{w}. \tag{1}$$

where $\chi(x) = 1$ if $x < 0$ and $\chi(x) = 0$ otherwise. w is the length of a time window centered at sample i. Basically, Con_i is equal to the proportion of the samples in the window that are closer than d_c to sample i. Cluster centers are recognized as samples for which the value of $\gamma_i = \delta_i \cdot \rho_i \cdot Con_i$ is particularly large.

2) Clustering: Now we have the cluster centers $\mathbf{C} = \{C_1, \ldots, C_{Nc}\}$, the next task is to cluster each sample s_i to these clusters. Assume that the locations of the samples follow the GMM (Gaussian Mixed Model) model. Therefore, we can estimate the probability that sample s_i belongs to C_k by:

$$pd_i^k = \beta \cdot P(C_k | I(i), Q(i)) \tag{2}$$

where $P(\cdot)$ is the probability density function of the Gaussian model, and β is a normalization constant. Then it seems that we can directly classify symbol i to C_k if pd_i^k is the maximum in $\mathbf{Pd}(i)$. However, directly classifying the samples located at the overlapping area (termed as *confused symbols*, as shown in Figure 8(c)) based on such method may incur errors. We propose to leverage the time domain information to address this problem.

In practice, the signal will dwell on a certain cluster between two transitions, thus successive symbols are likely to belong to the same cluster. We propose a parameter $\mathbf{Pt}(i) = \{pt_i^k \mid 1 \le k \le Nc\}$ to describe from time perspective that how likely symbol i belongs to cluster C_k. We have:

[1]The performance of LDBC is insensitive to the selected d_c [21].

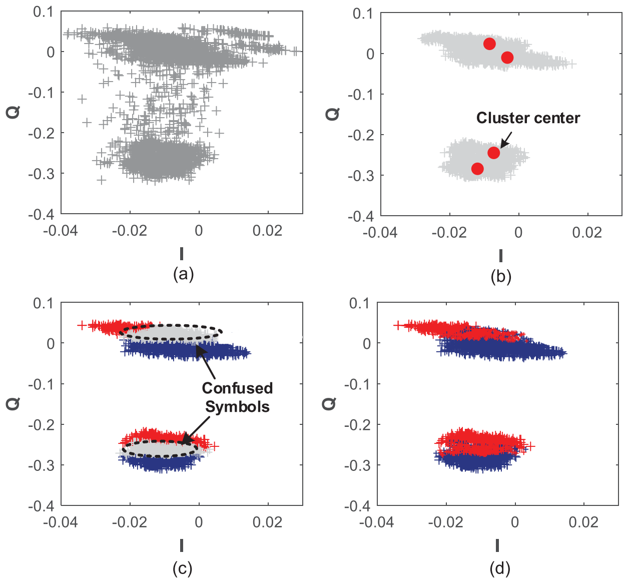


Fig. 8. Symbol clustering: (a) received samples. (b) identified cluster centers. (c) confused symbols. (d) clustering result.

$$pt_i^k = \frac{\#C_k}{w_c} \tag{3}$$

where $\#C_k$ denotes the number of samples that belong to cluster k in the time window centered at symbol i, and w_c is the length of the window, which is set according to the bit duration of the tags and the sampling rate of the reader. Now we can classify the confused symbols based on a joint consideration of $\mathbf{Pd}$ and $\mathbf{Pt}$. Specifically, we calculate the $\mathbf{P_{cluster}}(i) = \{pd_i^k \cdot pt_i^k \mid 1 \le k \le Nc\}$ for each symbol i, and classify symbol i to cluster C_k if $pd_i^k \cdot pt_i^k$ is the maximum in $\mathbf{P_{cluster}}(i)$. Algorithm 1 gives a high level overview on the clustering process.

3) Error Detection: Errors in sample clustering can seriously affect the subsequent decoding procedure. In the following, we explain how to detect and correct the errors in symbol clustering results. Here we only focus on the errors in cluster center identification. The errors in clustering each symbol (which may lead to bit errors) will be handled in Section VII.

Errors in cluster center identification are usually caused by i) treating multiple clusters as one cluster; or ii) treating one cluster as multiple cluster, both lead to incorrect number of clusters ($\neq 2^x$). But how to identify those false clusters? Indeed, false clusters exhibit some distinct features (such as the number of samples in the cluster and the area of the cluster), as make it possible to identify them.

Specifically, we propose a metric to describe how likely a cluster C_k is a false cluster:

$$PoE_{clu}(k) = \frac{R_{area}(k) + R_{num}(k)}{\sum_{sc(i)=C_k} Conf_{clu}(i)} \tag{4}$$

where $R_{area}(k)$ and $R_{num}(k)$ denote how much the area and the sample number of cluster C_k deviate from the corresponding average values. $Conf_{clu}(i) = max(\mathbf{P_{cluster}(i)})$ is the confidence of classifying sample s_i to C_k. When $PoE_{clu}(k) > 0$ a larger $|PoE_{clu}(k)|$ indicates a higher probability

Algorithm 1 Sample Clustering.

```
Require:
1: $N$ signal samples: $\mathbf{S} = \{s_1, \ldots, s_N\}$;
2: for $i = 1\ toN$ do
3:     $\rho(i)$ =DensCalculate($s_i$);
4:     $\delta(i)$ =DistCalculate($s_i$);
5:     $\gamma(i) = \rho(i) \cdot \delta(i)$;
6: end for
7: $\mathbf{C}$ =CenterIdentify($\gamma$);
8: for $i = 1\ toN$ do
9:     $\mathbf{Pd}(i)$ =CalculatePd($s_i$,$\mathbf{C}$);
10:    $Candi$ =Find($\mathbf{Pd}(i) > 0.4$);
11:    if length($Candi$)== 1 then
12:        $sc(i) = Candi$;
13:        $Fconf(i) = 0$;
14:    else
15:        $Fconf(i) = 1$;
16:    end if
17: end for
18: for  do$i = 1\ toN$
19:    if $Fconf(i) == 1$ then
20:        $\mathbf{Pt}(i)$ =CalculatePt($s_i$,$\mathbf{C}$);
21:        $\mathbf{P_{cluster}}(i) = \mathbf{Pd}(i) \cdot \mathbf{Pt}(i)$;
22:        $sc(i) = Find(max(\mathbf{P_{cluster}}(i)))$;
23:    end if
24: end for
```

Algorithm 2 Error Detection and Correction

```
Require:
1: The cluster centers: $\mathbf{C} = \{C_1, \ldots, C_k\}$;
2: The clustering result: $\mathbf{SC} = \{sc_1, \ldots, sc_N\}$;
3: Restriction of the time of corrections: $round_{th}$;
4: Search radius: $R_{th}$;
5: while $log_2|\mathbf{C}|$ is not an integer or $round < round_{th}$ do
6:     for $k = 1\ toK$ do
7:         $PoE_{clu}(k)$ =PoECalculation($C_k$);
8:     end for
9:     $C_m = argmax(|PoE_{clu}|)$;
10:    $S_m = argsc(i) = C_m$;
11:    $\gamma_m = \{\lambda_i | \lambda_i \in S_m\}$;
12:    if $PoE_{clu}(m) > 0$ then
13:        $\mathbf{C_m^{sp}}$ =FindSecondLargest($\gamma_m$);
14:        $\mathbf{SC_m^{sp}}$ =SampleClustering($S_m, C_m^{sp}$);
15:        $\mathbf{C} \leftarrow \mathbf{SC_m^{sp}}$;
16:        $K = K + 1$;
17:    else
18:        $\mathbf{C_{candi}}$ =FindCandi($C_m, R_{th}, \mathbf{C}$);
19:        $\mathbf{PoE_{candi}} = \{PoE_{clu}(k) | C_k \in \mathbf{C_{candi}}\}$;
20:        $C_{mate} = argmin(\mathbf{PoE_{candi}})$
21:        Merge($C_m$,$C_{mate}$);
22:        $K = K - 1$;
23:    end if
24:    $round = round + 1$;
25: end while
```

that C_k is a combined cluster; while when $PoE_{clu}(k) < 0$ a larger $|PoE_{clu}(k)|$ indicates a higher probability that C_k is an incomplete cluster.

4) Error Correction: We first calculate PoE_{clu} for each cluster, after which we correct the cluster which exhibits the largest $|PoE_{clu}|$. Specifically, if $PoE_{clu}(k) > 0$, we split this cluster; otherwise we merge this cluster with another cluster. After each correction, we update the PoE_{clu} of each cluster. We repeat the above process until the number of clusters is the power of 2.

- *Cluster splitting.* We identify the sample which exhibit the second largest γ_i in cluster C_k as the additional cluster center. Then we re-cluster the samples in C_k to this two centers using Algorithm 1 (line 7-23).
- *Cluster merging.* We identify the cluster whose distance to C_k is shorter than a threshold R_{th} and exhibits the minimum PoE_{clu} as the mate-cluster of C_k. Then we merge C_k with its mate-cluster.

A threshold is set to limit the times of cluster corrections. When the threshold is exceeded, we stop the error correction process. Note that FlipTracer can still decode the signal from some of the tags, achieving partial decoding, even the combined clusters are not identified by the error correction method. Algorithm 2 gives a high level overview on the error detection and correction process.

B. Building Connections in the OFG

Now we have the symbol clusters (the nodes in the OFG), and the next step is to build the connections, namely, to form the neighbor relationships between clusters. Based on our observations in Section III, the transition probabilities between neighbor clusters are significantly higher than those between non-neighbor clusters. Thus we can recognize neighbor clusters based on the transition probabilities between clusters.

We first calculate the transition probabilities between clusters. For each cluster $C_i \in \mathbf{C}$, we can calculate the transition probability between C_i and every other cluster C_j as:

$$P_{trans}(C_i, C_j) = \frac{\#(C_i \leftrightarrow C_j)}{\sum_{C_k \in \mathbf{C}} \#(C_i \leftrightarrow C_k)} \tag{5}$$

where $\#(C_i \leftrightarrow C_j)$ denotes the number of transitions between C_i and C_j, and $\sum_{C_k \in \mathbf{C}}(C_i \leftrightarrow C_k)$ denotes the total number of transitions between C_i and all the other clusters in $\mathbf{C}$.

In a N-tag collision case, each cluster C_i has n neighbors, denoted as $\mathbf{C_{nei}}(C_i)$. Theoretically, the N neighbors of C_i should be the N clusters leading to the N highest $P_{trans}(C_i, C_j)$, termed as *potential neighbors* of C_i (denoted as $\mathbf{C_{pos}}(C_i)$). In practice, however, burst noise may result in wrong cluster labels, bringing deviations to the measured transition probabilities. An extreme case is that, for a cluster C_i, the measured transition probabilities between C_i and its non-neighbor clusters might be even higher than that between C_i and its neighbors. Thus, directly identifying $\mathbf{C_{pos}}(C_i)$ as $\mathbf{C_{nei}}(C_i)$ may incur errors. To solve this problem, we propose a new metric $Conf(C_i)$ for each cluster C_i to describe the *confidence* for identifying $\mathbf{C_{pos}}(C_i)$ as C_i's neighbors:

$$Conf(C_i) = \frac{min\{P_{trans}(C_i, \mathbf{C_{pos}}(C_i))\}}{max\{P_{trans}(C_i, \mathbf{C_{nopos}}(C_i))\}} \tag{6}$$

where $\mathbf{C_{nopos}}(C_i)$ denotes the clusters that do not belong to $\mathbf{C_{pos}}(C_i)$. A larger gap between $P_{trans}(C_i, \mathbf{C_{pos}})$ and $P_{trans}(C_i, \mathbf{C_{nopos}})$ indicates a higher confidence.

Algorithm 3 Connection Building.

```
Require:
1: The cluster centers: $\mathbf{C} = \{C_1, \ldots, C_k\}$;
2: The clustering result: $\mathbf{SC} = \{sc_1, \ldots, sc_N\}$;
3: $\mathbf{P_{Trans}}$ =CalPtrans($\mathbf{SC}$);
4: for $i = 1$ to $K$ do
5:     $\mathbf{C_{pos}}(C_i)$ =FindPotentialNeighbour($C_i$,$K$);
6:     $Conf(C_i) = \frac{min\{P_{Trans}(C_i, \mathbf{C_{pos}}(C_i))\}}{max\{P_{Trans}(C_i, \mathbf{C_{nopos}}(C_i))\}}$;
7: end for
8: $\mathbf{C_{ord}}$ =DescendOrder($\mathbf{C}, \mathbf{Conf}$);
9: for $i = 1$ to $K$ do
10:     $\mathbf{P_{Torder}}(C_i)$ =DescendOrder($\mathbf{C}, \mathbf{P_{Trans}}(C_i)$);
11:     $j = 1$;
12:     while $\mathbf{Nei_{num}}(C_i) < N$ do
13:         if OddDet($E \leftarrow \langle C_i, \mathbf{P_{Torder}}(C_i)\rangle$)== 0 then
14:             $E \leftarrow \langle C_i, \mathbf{P_{Torder}}(C_i)\rangle$;
15:             $\mathbf{Nei_{num}}(C_i) + +$;
16:             $\mathbf{Nei_{num}}(\mathbf{P_{Torder}}(C_i)) + +$;
17:             $j + +$;
18:         end if
19:     end while
20: end for
```

To improve neighbor identification robustness, we propose to first sort the clusters in descending order of their $Conf$, and identify the neighbors for the clusters sequentially. This allows us to process the clusters that have the higher $Conf$ earlier. Specifically, for the k-th cluster, if m of its N neighbors have been covered by the previous $k-1$ clusters (which have higher $Conf$), we only need to identify its other $(N-m)$ neighbors based on P_{trans}. Algorithm 3 gives a high level overview on the connection building process.

C. Correcting False Connections

Now we have both the symbol clusters (nodes in the OFG) and the connections between neighbor clusters, so we can get the OFG. However, one problem we face here is how to handle the false connections (the connections which connect two non-neighbor clusters) in the OFG. Although it is a fairly rare occurrence, when it occurs, the following cluster identification process will be seriously affected. Thus we propose a false-tolerance design to avoid forming such false connections when constructing the OFG.

We propose such a design based on our intuition that the false connections will lead to abnormal structures in the OFG. In practice, *if all the connections in the OFG are correctly formed, there will be no odd-node loops in the OFG.* The reason is that for a combined state $[S_1, \ldots, S_N]$, its each state S_i has to experience two transitions to transfer to its initial state (i.e., $S_i \rightarrow \overline{S_i} \rightarrow S_i$). Thus the combined state $[S_1, \ldots, S_N]$ has to experience an even number of transitions to transfer to the initial combined state. Figure 9(a) shows an example of the loop with four nodes.

However, a false connection will inevitably form a loop with an odd number of nodes, which can be exploited as an indicator of errors. As an example, consider a false connection $< C_i, C_j >$ connecting two clusters whose combined states have two different states (denoted as States S_p and S_q). Thus, the combined states of C_i and C_j will have at least one

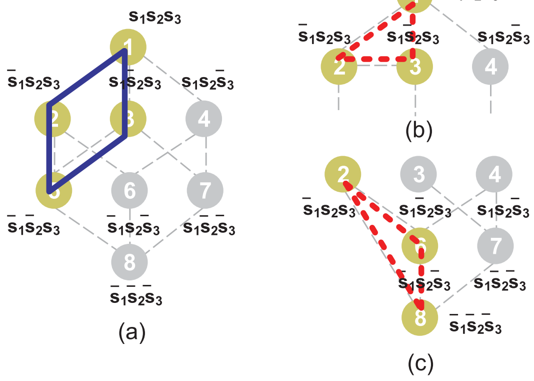


Fig. 9. Loops in the OFG of a 3-tag collision case: (a) an example of a correct loop. (b) and (c) are two examples of the abnormal loops.

common neighbor cluster (denoted as C_k) which have one different state (S_p or S_q) with C_i and C_j. Clusters C_i, C_j, and C_k form a 3-node loop. Figures 9(b) and 9(c) show two examples of such abnormal loops. But how to find such *abnormal loops*?

An intuitive method is to traverse the graph, which however leads to high computation complexity. The method used in FlipTracer is to check whether C_i and C_j have common neighbors (i.e., whether $< C_i, C_j >$ forms a loop with three nodes). If it does, we will label C_j as a non-neighbor cluster of C_i.[2] Why would this work?

Indeed, since the transition probability between clusters decreases with the number of different states between them, FlipTracer can only mistake two non-neighbor clusters which have two different states as neighbor clusters. Clusters which have more than two different states between them are unlikely to be identified as neighbors, because the transition probability between them is too low. As discussed earlier, if two clusters C_i and C_j have two different states between them, the connection $< C_i, C_j >$ will inevitably form a loop with 3 nodes. Therefore, the lightweight method mentioned above can efficiently identify almost all false connections.

At last, note that "odd number of nodes in a loop (i.e., the abnormal loop)" is a sufficient condition for false connections but not a necessary condition. Therefore, we cannot identify all the false connections by using the above method. A potential method to enable more robust error detection is to check *whether the constructed graph has the valid structure of an OFG*. Here the valid structure is described as follows: for a cluster C_i located at layer l, it must have l parent clusters at layer $l-1$, and $N-l$ child clusters at layer $l+1$. The probability for an invalid OFG exhibiting such a valid structure is extremely low. To say at the least, even when individual false connections occurs, FlipTracer can still decode the signals from some of the tags, achieving partial decoding.

VI. Cluster Identification

In this section, we introduce how to identify the combined state of each cluster by tracking the OFG. We assume that there is an anchor cluster, whose combined state is known. This is reasonable because we can always identify the cluster representing all "L" states. Specifically, when all the tags are being charged, there is only one cluster on the IQ domain,

[2]Note that this process assumes the connects that formed earlier (which exhibit higher $Conf$) are more likely to be correct.

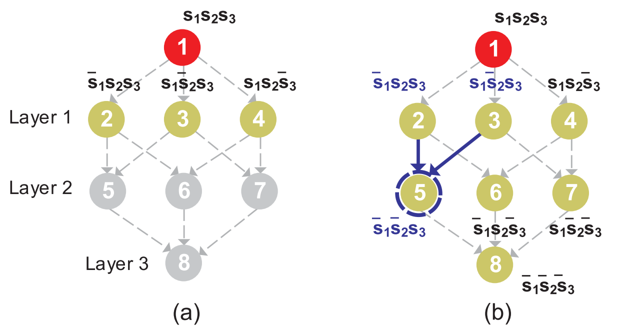


Fig. 10. The process of decoding a 3-tag collision: (a) Identify the first layer clusters. (b) Identify the clusters in Layer $l-1$.

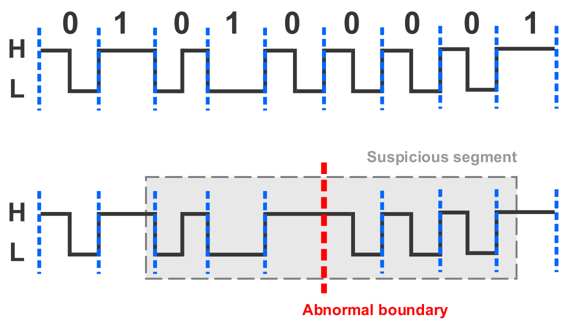


Fig. 11. FM0 coding: (a) A correct state sequence. (b) A state sequence with errors.

which is the all "L" cluster. We can always identify the all "L" cluster since its location is relatively stable.

We first layer the OFG. Specifically, the anchor cluster is the root of the OFG, denoted as C_{root}. We define the neighbors of C_{root} as its child nodes, which form the first layer of the OFG. For each Layer l cluster (denoted as C_l^i, and $l > 0$), we define its neighbors, that is not in Layer $(l-1)$, as its child nodes. The child nodes of the Layer l clusters form the $(l+1)$-th Layer of the OFG. Then we can identify the clusters in the OFG layer by layer, starting from C_{root}.

Claim 1 (We Can Identify the Combined States of the Layer-1 Clusters Based on the Combined State of C_{root}): If the combined state of C_{root} is $[S_1, S_2, \ldots, S_N]$, we can identify its neighbor clusters (i.e., the Layer 1 clusters) as $[\overline{S_1}, S_2, \ldots, S_N]$, $[S_1, \overline{S_2}, \ldots, S_N]$, ..., $[S_1, S_2, \ldots, \overline{S_N}]$. Figure 10(a) gives an example of the above process, in a 3-tag collision case. In this example, if we denote the tags that cause transitions between Clusters 1 and 2, 1 and 3, 1 and 4 as Tag 1, Tag 2, and Tag 3, respectively,[3] the Clusters 2, 3, and 4 can be identified as $[\overline{S_1}, S_2, S_3]$, $[S_1, \overline{S_2}, S_3]$, and $[S_1, S_2, \overline{S_3}]$, respectively. We will explain the above points in the appendix with a concrete example.

Claim 2 (There Are l Different States Between the Combined States of C_{root} and Each Layer l cluster): We define the distance between two clusters C_a and C_b as the number of different states between their combined states, denoted as $d(C_a, C_b)$. Obviously, the distance between C_{root} and each Layer 1 cluster C_1^i is 1. Denote the k-th child node of C_1^i as $Ch(C_1^i, k)$, we have i) $d(Ch(C_1^i, k), C_1^i) = 1$, and ii) $d(Ch(C_1^i, k), C_{root}) > d(C_1^i, C_{root}) = 1$. Thus, we can derive that $d(Ch(C_1^i, k), C_{root}) = d(C_1^i, C_{root}) + 1 = 2$, namely $d(C_2^i, C_{root}) = 2$. Iteratively, we can derive that $d(C_l^i, C_{root}) = l$.

Now we can provide a set of candidate combined states for the Layer l $(l > 1)$ clusters. Taking the 3-tag collision case in Figure 10 as an example, we can provide a set of candidate combined states for the Layer 2 clusters as $[\overline{S_1}, \overline{S_2}, S_3]$, $[\overline{S_1}, S_2, \overline{S_3}]$ and $[S_1, \overline{S_2}, \overline{S_3}]$. The next task is to map each combined state in the candidate set to each cluster.

Claim 3 (We Can Identify the Combined States of the Layer-l Clusters Based on Those of the Layer-$(l-1)$ Clusters): As discussed in Claim 2, the distance between C_l^i and C_{root} is l. Thus, there are l different states between the combined states of C_{root} and C_l^i. We denote these states as $S_{k_1}, S_{k_2}, \ldots, S_{k_l}$, where $k_1, k_2, \ldots, k_l$ are the indexes of the states. Its easy to understand that C_l^i have l fathers in Layer $(l-1)$, and the different states between C_{root} and its fathers are $\{[S_{k_2}, S_{k_3}, \ldots, S_{k_{l-1}}, S_{k_l}], [S_{k_1}, S_{k_3}, \ldots, S_{k_{l-1}}, S_{k_l}], \ldots, [S_{k_1}, S_{k_2}, \ldots, S_{k_{l-2}}, S_{k_{l-1}}]\}$. Thus if the combined states of the Layer-$(l-1)$ clusters have been identified (which means that $k_1, k_2, \ldots, k_l$ are known), the combined state of C_l^i can be identified accordingly. As the example shown in Figure 10(b), the combined state of Cluster 5 can be identified as $[\overline{S_1}, \overline{S_2}, S_3]$ if Clusters 2 and 3 are identified as $[\overline{S_1}, S_2, S_3]$ and $[S_1, \overline{S_2}, S_3]$, respectively.

In summary, the process of cluster identification is:

- identify the combined state of an anchor cluster;
- identify the Layer 1 clusters based on Claim 1;
- identify the clusters in Layer 2$\sim N$ based on Claim 3;

Then the combined states of all the clusters are identified.

VII. Reliable Decoding

Now the combined states of all the clusters are identified. By examining those combined states, FlipTracer outputs a sequence of "H" and "L" states to represents the transmitted signal for each tag. The sequence can be decoded using the conventional single tag decoder.

Unlike the single tag transmission scenario, however, in the collision scenario, wrong labels in the symbol clustering process may increase the BER (bit error rate) of individual tags. Thus, we would like to correct the errors in the state sequence before inputting it to the decoder. The predictable state flipping pattern of the standardized FM0 and Miller codes suggests a possible solution - we can simply leverage the fact that certain state sequences are not possible.

FM0/Miller code inevitably inverts the signal state at every bit boundary (as shown in Figure 11(a)). Thus it is a obvious error if two samples on the opposite sides of a bit boundary have the same state. We term such a bit boundary as an *abnormal bit boundary* and the state sequence with a length of K chips ($K = 6$ in our implementation) centered at this abnormal boundary as a *suspicious segment* (as shown in Figure 11(b)). Once we detect an abnormal bit boundary, we use a Conditional Random Field based decoder to correct the errors in the corresponding suspicious segment.

For a tag i, suppose that we detect a suspicious segment with a sequence of $k = 1, 2, \ldots, K$ chips in the state sequence. For each chip, we introduce a random variable S_k ($S_k = H$ or L) to represent its state. The joint probability of assigning a sequence of states $\{S_1, S_2, \ldots, S_K\}$ to all the chips $k = 1, 2, \ldots, K$ is given as follows:

$$P(\mathbf{S}) = \beta \prod_{k} \phi(S_k) \prod_{k,k+1} \psi(S_k, S_{k+1}) \tag{7}$$

[3]Such IDs are the "temporary IDs" which are used to distinguish (rather than to denote) the signals from different tags. What we essentially care about are the IDs (or data) embedded in the payloads of the packets.

where β is a normalization constant. The factor $\phi(S_k)$ is the emission probability of assigning S_k to k, which can be given by the $\mathbf{P_{cluster}}$ of the symbols in $\mathbf{C_{S_k}}$ ($\mathbf{C_{S_k}}$ represents the clusters indicating a S_k state of tag i). The factor $\psi(S_k, S_{k+1})$ is the transition probability between S_k and S_{k+1}. Specifically, if the transition between S_k and S_{k+1} indicates a bit boundary, we have $\psi(S_k, S_{k+1}) = 0$ if $S_{k+1} = S_k$, and $\psi(S_k, S_{k+1}) = 1$ if $S_{k+1} = \overline{S}_k$. Otherwise, we have $\psi(S_k, S_{k+1}) = 0.5$. We calculate $P(\mathbf{S})$ for all possible state assignments and identify the one which the maximum $P(\mathbf{S})$ as the correct one.

VIII. FlipTracer for Tag Identification

In this Section, we introduce how FlipTracer can be exploited to maximize the throughput of the EPC protocol [22], [23]. In the EPC protocol, the reader divides time into N_s slots (here $N_s = 2^Q$, and Q is a parameter to control the number of time slots). Each tag randomly selects a slot for data transmission. To eliminate collisions, the reader has to make the number of slots fairly large, which reduces efficiency since many empty slots are produced.

FlipTracer solves this problem since it allows multiple tags to transmit concurrently, which further decreases the number of required slots. But how to set the number of time slots, so that the throughput can be maximized? Indeed, the throughput is determined by two factors: i) packet reception ratio (PRR) for different number of colliding tags k ($1 \leq k \leq N_{tag}$, where N_{tag} is the total number of tags), denoted by PRR_k; and ii) the distribution of k, i.e., the proportion of the slots that involve k colliding tags, denoted by Col_k. Col_k is further affected by N_{tag} and N_s. Suppose that all the tags transmit at the same bit rate B, then we have the average throughput as:

$$Th(N_s) = \frac{\sum_{k=1}^{N_{tag}} Col_k \cdot k \cdot B \cdot PRR_k}{N_s}, \tag{8}$$

and the average PRR as:

$$PRR(N_s) = \frac{\sum_{k=1}^{N_{tag}} Col_k \cdot PRR_k}{N_s}. \tag{9}$$

For each N_{tag}, if we can estimate $Th(N_s)$ and $PRR(N_s)$, we can get the N_s which maximizes the throughput.

To this end, we should first estimate: i) the number of the tags N_{tag}; ii) the bit rate of the tags B; iii) the concurrency level Col_k; and iv) the PRRs for different numbers of colliding tags PRR_k. Since B is known by the reader and N_{tag} can be obtained through cardinality estimation methods [24], our target is to estimate Col_k and PRR_k.

A. Estimation of Col_k

Suppose that a tag will select each slot with a probability of $\frac{1}{N_s}$. For each slot i, the probability that k tags select it for data transmission can be estimated as:

$$P(SLOT_i = k) = C_{N_{tag}}^k \cdot \left(\frac{1}{N_s}\right)^k \cdot \left(1 - \frac{1}{N_s}\right)^{N_{tag}-k}, \tag{10}$$

Therefore, the number of slots which involve k tags can be estimated as:

$$\begin{aligned} Col_k &= \sum_{i=1}^{N_s} P(SLOT_i = k) \\ &= N_s \cdot C_{N_{tag}}^k \cdot \left(\frac{1}{N_s}\right)^k \cdot \left(1 - \frac{1}{N_s}\right)^{N_{tag}-k}, \end{aligned} \tag{11}$$

B. Estimation of $\mathbf{PRR_k}$

PRR_k will change with environmental factors like interference and noise. For tracking the PRRs, the reader constructs a *PRRMap* to maintain PRRs for different number of colliding tags. Initially, the *PRRMap* is empty. We select an initial Q, and get an initial N_s accordingly. After several rounds of data transmissions, we get the PRRs for different k and add them to the *PRRMap*. Then we can select Q according to the *PRRMap*. We also update *PRRMap* after it is constructed. Specifically, in each new round of transmissions, the reader will get new PRRs under different k, and update the corresponding items in the *PRRMap*.

IX. Analysis

A. Computation Overhead and Latency

The computation overhead is dominated by the clustering, OFG construction and cluster identification components.

In the symbol clustering component, to reduce the input size for faster clustering, FlipTracer aggregates received symbols into grids and cluster those grids [7]. This reduce the computation overhead to $O(N_g \cdot logN_g + M)$, where N_g is the number of grids and M is the number of confused samples (the average value of M is 80 according to our experiments). Let N be the number of the tags. The computation overheads of the OFG construction and cluster identification components are $O(N^2 + 2^{N-1} \cdot N)$ and $O(2^{N-1} \cdot N)$, respectively. Since the sampling capacity of the reader is limited, the aggregated throughput cannot increase infinitely with the number of tags. We find through experiments that the performance of FlipTracer peaks at $N = 5$.

B. Impact of Channel Variation

Dynamic working environment leads to channel variation, which changes the locations of the clusters on the IQ domain. FlipTracer is robust to such dynamics. Specifically, FlipTracer splits the signal into 5ms windows, and creates an OFG for the signal samples in each window. Such a short time window is usually within the channel coherence time [7]. The reasons are explained as follows:

The first factor that may lead to channel variation is the mobility of the tag. Assume that the tag moves at a speed of 1m/s (this speed is larger than the claimed upper-bound moving speed of the tags in most RFID applications). The resulting displacement of the tag during 5ms is only 5mm. According to the path loss model of signal, a 5mm displacement only results in a less than 0.1dBm variation in signal strength. Moreover, since the wavelength is 160mm, a 5mm displacement only results in a less than 0.2rad variation in phase. In total, the resulting signal displacement in the IQ domain is less than 0.002, which is even smaller than the noise level (0.003 according to our experimental results). Therefore, FlipTracer is robust to the mobility of the tags.

Another factor that may lead to channel variation is the change in the environment, e.g., obstacle's movement between the reader and the tags. We have conducted an experiment to observe how the obstacle affects the signal's locations on the IQ domain. In the experiment, we use a 30×23×5cm book to block the transmission path between a reader and a tag. We move the book with the speed about 1m/s. The results show that in this case, the signal displacement in the IQ domain during 5ms is only 0.0015, which is also smaller than the noise level.

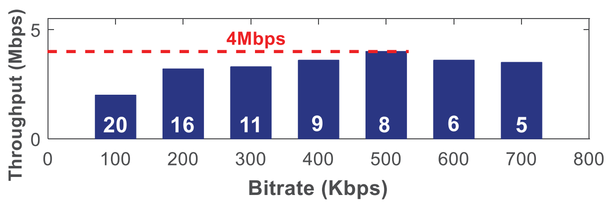


Fig. 12. Maximum throughput vs. bitrates.

C. Decoding Capacity of FlipTracer

1) Impact of the Bitrate: Theoretically, the decoding capacity of FlipTracer is highly determined by the bitrate of the tags. Specifically, since the sampling capacity of the reader is limited, when the bitrate of the tags increases, signal edges of different tags may align with each other with a high probability. As noted earlier, for a certain cluster C_i, if the transition probabilities between C_i and its neighbors are higher than that between C_i and other clusters, we can correctly find all the neighbors for C_i. Otherwise, the decoding will fail. Assume that N tags transmit concurrently with the same bitrate B and the sampling rate of the reader is S. The transition probability between C_i and another cluster C_j ($d(C_i, C_j) = dis$),[4] can be calculated as follows:

$$P(C_i, dis) = \frac{\binom{\frac{S}{B\cdot e}}{1} \cdot \binom{\frac{S}{B\cdot e}-1}{N-dis} \cdot (N-dis)!}{\frac{S}{B\cdot e}^{N}}, \tag{12}$$

where e is the width of the signal edge. $P(C_i, 1) > P(C_i, dis)$ and $dis \geq 2$ for each C_i ensures correctly constructing the OFG. Figure 12 shows the numerical results about the maximum throughput of FlipTracer under different bitrates, where the number marked on each bar means the corresponding number of concurrent tags. Figure 12 presents the theoretical upper bound of the throughput that FlipTracer can achieve, i.e., 4Mbps (concurrent transmissions from 8 tags at 500Kbps).

2) Impact of Channel Quality: In practice, channel quality can also affect the performance of FlipTracer. Specifically, when channel quality decreases, signal clusters may seriously overlap with each other. In this scenario, FlipTracer may identify multiple clusters as one cluster, making sample clustering more difficult and error prone. Experimental results shown in Section X tells that the throughput of FlipTracer peaks when 5 tags transmit concurrently. Although the capacity of FlipTracer is limited it can still significantly improve the performance of the standard EPC protocol (more details in Section X).

3) Potential Ways to Improve the Capacity of FlipTracer: As discussed above, the decoding capacity of FlipTracer is limited due to the quality of the channel and the limited sampling rate of the reader. We will address this issue in our future work. Potential solution may include the following idea: i)We may exploit more fine-grained characteristics to enhance the efficiency and accuracy of clustering. ii)We may have the tags transmit through different carriers (i.e. different channels), which is also a potential solution to improve the scalability.

D. Performance of the FlipTracer+EPC Protocol

As we have discussed in Section VIII, given i) the number of the tags N_{tag}, which can be obtained through cardinality estimation methods; and ii) the decoding capacity of FlipTracer, which is discussed in Section IX-C, the throughput bound of FlipTracer+EPC is highly related to *the number of slots* N_s. Figure 13 shows the throughput of FlipTracer and the conventional single tag decoder under different N_s to N_{tag} ratio, which is estimated based on Equations (8) and (12).

[4]Note that $d(C_i, C_j) = dis$ means the transition between C_i and C_j is caused by the aligned edges from dis tags. Clearly, $d(C_i, C_j) = 1$ means that C_i and C_j are neighbors.

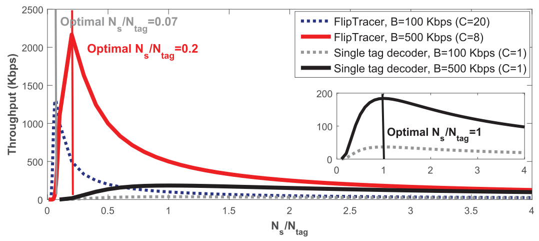


Fig. 13. Expected throughput of FlipTracer+EPC, depending on the N_s to N_{tag} ratio.

We can see that for the single tag decoder, the maximum throughput is achieved if the number of slot is set equal to the number of tags. In comparison, due to the ability of parallel decoding, FlipTracer exploits a much smaller number of slots for maximum throughput. Specifically, the maximum throughput of FlipTracer are 1.3 Mbps and 2.1 Mbps when the bitrates of the tags are set at 100 Kbps and 500 Kbps, respectively, and the corresponding throughput-optimal number of slots in those two cases are $0.07N_{tag}$ and $0.2N_{tag}$, respectively.

X. Evaluation

A. Experiment Setting

1) USRP Reader: FlipTracer is built based on the USRP N210 software radio reader with UBX RF daughterboards and two 900 MHz circular antennas. By default, the sampling rate is set at 20MHz, and the transmission power is 20dBm.

2) Backscatter Tags: We implement FlipTracer on programmable WISP tags. Implementation of FlipTracer only requires a slight modification to the EPC protocol. Specifically, we just remove the elements for Slot Aloha operation and thus the tags will respond concurrently. The packets are encoded with FM0 encoding scheme. The default bitrate is set as 100Kbps and the default packet length is set as 100 bits.

In the experiment, the default distance between the reader and the tags is 2 feet, and the default distance between the tags is 0.5 feet. Note that we require such a short distance between the reader and the tags because the communication range of the programmable WISP tags that used in our experiments is much lower than that of the COTS tags.[5]

We compare FlipTracer with the following two schemes:

- **Cluster-based parallel decoding (CB).** This approach assumes that tags' channel coefficients are static and can be linearly combined when collision occurs.
- **BiGroup.** BiGroup [7] assumes that the signal edges of a tag come with a relatively fixed interval. It extracts the sequence of each tag's signal transitions in the time domain, and connects with the symbol clusters in the IQ plane for parallel decoding.

B. Evaluation of OFG Construction

We first focus on the OFG construction component which has a significant impact on the overall performance.

[5]The SNR of the tag that used in our experiment decreases to 4dB when the distance between the reader and the tag increases to 6 feet.

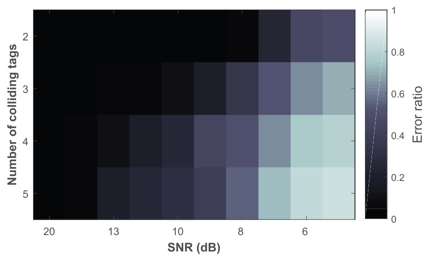


Fig. 14. Error ratio of the OFG construction component.

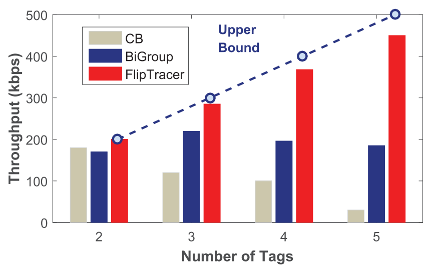


Fig. 15. Aggregated throughput with different numbers of tags.

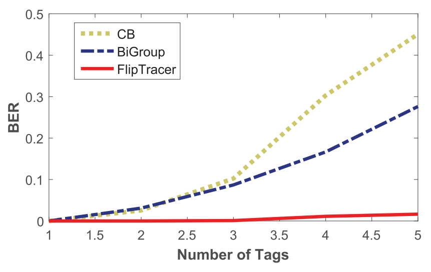


Fig. 16. BERs with different numbers of colliding tags.

Two factors can affect the performance of this component - the number of concurrent tags and the SNR. In the experiments, we increase the number of tags from 2 to 5. For each tag number, we change the SNR by adjusting the distance between the reader and the tags. Specifically, we change the distance from 1 to 6 feet, during which the SNR of the tags decreases from 20 dB to 4 dB. We collect 100 collided packets for each setting and show the corresponding error ratio of OFG construction in Figure 14.

As shown in the figure, the error ratio is low under most settings, but increases when the SNR decreases. The reason is that when the SNR is low, signal clusters may overlap with each other, making sample clustering more error prone.

C. FlipTracer Goodput

1) Aggregated Throughput: Figure 15 compares the aggregated throughput achieved by different schemes for different numbers of tags. We can see that the throughput of FlipTracer is very close to the maximum possible throughput in all cases. The throughput gain of FlipTracer is big: in the 5-tag collision case, FlipTracer is 2.5× better than BiGroup and 15× better than CB.

2) BER: Figure 16 plots the BERs of different schemes. We see that FlipTracer greatly outperforms CB and BiGroup. In the 5-tag collision case, the BER of FlipTracer is 28× lower than that of CB and 17× lower than that of BiGroup. We also find that for all three schemes, the BERs rise rapidly

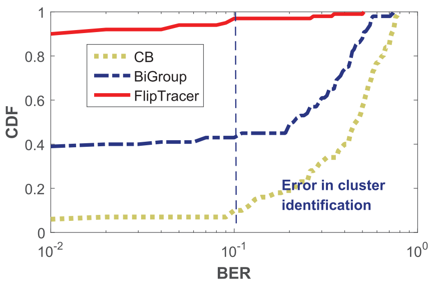


Fig. 17. CDFs of BERs for different schemes when the number of tag is 5.

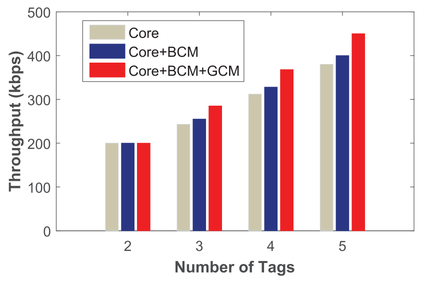


Fig. 18. Breakdown of each component's contribution to throughput.

if the number of tags is larger than 4. This is because the performance of cluster identification decreases with the number of tags, which leads to excessive bit error in the decoding process.

To further investigate the major reason of the bit errors, in Figure 17 we plot the CDFs of BERs for the three schemes in the 5-tag collision case. The BERs above 0.1 are caused by errors in the cluster identification process, which generally results in excessive bit errors. We can see that the cluster identification error of FlipTracer is much lower than that of BiGroup and CB.

3) Breaking Down the Benefits: Let us now break down the aggregated throughput by different components of our design to see how much each of them matters. Figure 18 shows the result. We start with the approach that only uses the core design of FlipTracer.[6] Then we add bit error correction module (BCM). Finally we add the OFG correction module (GCM).

We see that each of these benefits the overall throughput. For example, when the number of the colliding tag is 5, the core design of FlipTracer causes about 20% of throughput reduction, compared to what is achieved by FlipTracer. Bit error correction component improves the throughput by about 5% and adding the OFG correction component further improves it by another 15%. We can see that the OFG correction component leads to more performance gain than the bit error correction component.

D. Impacts of Bitrate

In this experiment, we vary the bitrate of the 5 tags from 100Kbps to 600Kbps. Since the WISP platform currently support only a 256Kbps bitrate, we conduct simulations to see the performance of FlipTracer with 500∼600 Kbps bitrate. Figure 19 shows that the aggregated throughput crashes at 600Kbps. This gives us an empirical upper bound on the throughput FlipTracer can support - **concurrent transfer from**

[6]FlipTracer that is not integrated with the OFG correction module and the bit error correction module

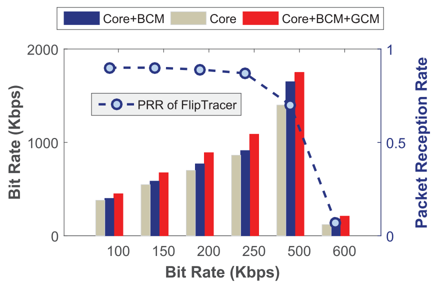


Fig. 19. Throughput under different bitrates.

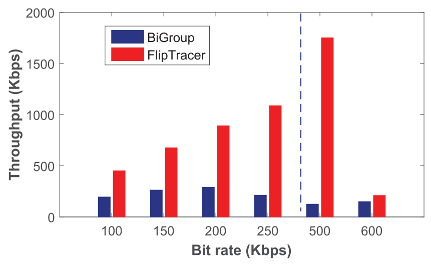


Fig. 20. Throughput comparison under different bitrates.

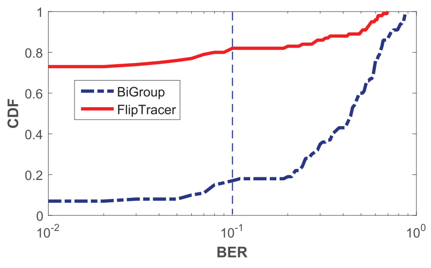


Fig. 21. BERs when bitrate is 500Kbps and the number of tag is 5.

5 tags at 500Kbps. We find that the empirical upper bound is lower than the theoretical upper bound. The reasons behind are: i) the limited channel quality affects the resolution of sample clustering in practice; and ii) signal edges are actually unevenly distributed in the time domain, which leads to higher probability of edge alignment.

In Figure 20, We compare the aggregated throughput of 5 concurrent tags achieved by BiGroup and FlipTracer (since CB can not support more than 4 tags) under different bitrates. We observe that the throughput of BiGroup crashes after 200Kbps. The reason is that a high bitrate will make the signal edges from different tags be too close to each other. Given that the bit duration of each tag is not stable, the reader can hardly distinguish the edges of different tags. This results in excessive errors. Figure 20 also shows that the maximum throughput of BiGroup is 300Kbps. Compared with BiGroup, FlipTracer is more robust to the high bitrate: when tags transmit at 500Kbps, the throughput of FlipTracer is $14\times$ higher than that of BiGroup.

In Figure 21, we plot the CDFs of BERs for FlipTracer and BiGroup when the bitrate is 500Kbps and the number of tags is 5. By comparing Figures 17 and 21, we find that a high bitrate can seriously affect the cluster identification accuracy of BiGroup, which results in the throughput crash of BiGroup after 200 Kbps.

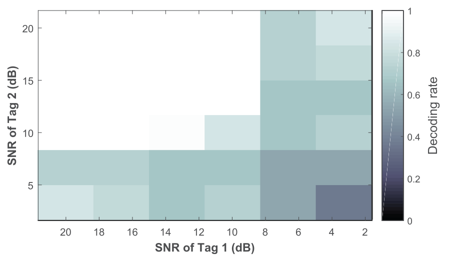


Fig. 22. Impact of poor channel condition.

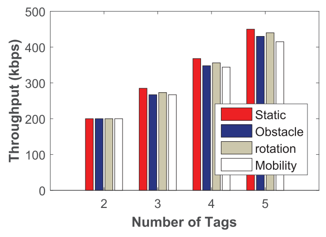


Fig. 23. Throughput of FlipTracer under different working conditions.

E. FlipTracer in "Bad" Working Condition

This subsection evaluates the performance of FlipTracer under some "bad" working conditions.

1) Poor Channel Quality: We first observe how the absolute SNR of each tag and the relative SNR between different tags affect the performance of FlipTracer. In the experiment, we use two tags. Both the absolute and the relative distances between the tags and reader are varied. Specifically, for each tag, we change the distance between the reader and the tags from 1 to 6 feet, during which the SNR of the tag decreases from 20 dB to 4 dB. This results in different absolute and relative SNRs of the two tags. The decoding rate of FlipTracer under different settings are shown in Figure 22.

The figure tells that: i) the decoding rate of FlipTracer is high under most settings, but decreases when the SNR decreases; ii) when the absolute SNR of one tag is fixed, the decoding rate does not strictly decrease with the decreased SNR of the other tag. For example, when the SNR of Tag 1 is 20 dB and that of Tag 2 decreases to 8dB, the decoding rate decreases since the distance between clusters "XH" and "XL" (where X = H or L) become shorter. Once the overlapping occurs, the signal of both tags cannot be correctly decoded and thus the decoding rate seriously decreases. However, when the SNR of Tag 2 decreases to 4dB, clusters "XH" and "XL" almost completely coincide with each other. That is to say, the signal of Tag 2 can be considered as noise. In this case, we can at lest correctly decode the signal from Tag 1, which leads to a increased decoding rate.

2) Dynamic Environment: Figure 23 shows the impact of dynamic working conditions. Four cases are compared: i) the stable working condition; ii) the tags and the reader are fixed, but there is a moving obstacle in the vicinity of the tags; iii) the reader is fixed but the locations of the tags keep changing; and iv) the reader is fixed but the tags' orientations keep changing.

We find in Figure 23 that the throughput of FlipTracer under dynamic scenarios is slightly lower than that in the static scenario. The reason behind is that although the obstacle, rotation and mobility will change the channel coefficients of

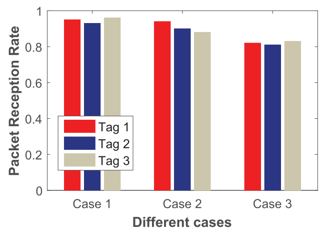


Fig. 24. Throughput of FlipTracer when tags transmit at different bitrates.

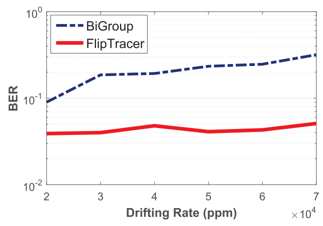


Fig. 25. Impact of clock drifting rate.

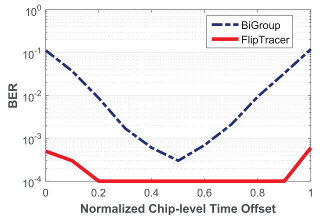


Fig. 26. Impact of delay offset.

the tags, FlipTracer is able to decode the collided signals without channel coefficient information.

3) Co-Existence of Fast and Slow Tags: FlipTracer can support widely different bitrates among tags. To evaluate this benefit, we use three tags and evaluate the aggregated throughput in three different scenarios: i) we let all the three tags transmit at 100Kbps; ii) we let the three tags transmit at 100Kbps, 150Kbps and 200Kbps, respectively. iii) we let two tags transmit at 50Kbps and the other one transmit at 250Kbps.

Figure 24 shows the packet reception rate achieved by each tag under different scenarios. The results show that: i) the throughput achieved in Case 2 is almost the same as that achieved in Case 1. ii) even in Case 3 where the bitrate of the fast tag is 5× as that of the slow tags, the throughput only slightly decreases.

F. Impact of Uncontrollable Factors

In order to understand the impacts of the uncontrollable parameters, we conduct extensive simulations.

1) Clock Drifting Rate of Tags: One of the key benefits of FlipTracer is that it's robust to high clock drifting rates of the tags. Figure 25 shows the BER comparison of BiGroup and FlipTracer in different clock drifting rates when the number of colliding tags is 5. We can see that the BER of BiGroup method increases rapidly when the drifting rate increases from 20,000ppm to 60,000ppm (the probable drifting rate of the COTS tags). In comparison, the increase in the BER of FlipTracer is insignificant.

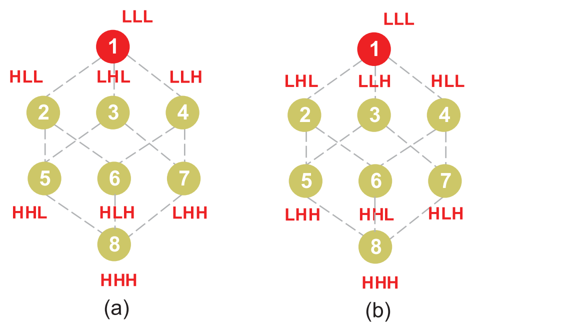


Fig. 27. Two different assignments of the Layer 1 clusters: (a) assignment Case 1; (b) assignment Case 2.

2) Delay Offset of Different Tags: Figure 26 shows the impact of delay offset. In this simulation, the number of colliding tags is 2. The X-axis presents the normalized delay offset. Clearly, for both schemes, the best BER is achieved when the normalized delay offset is 0.5, where the signal edges of different tags are far away from each other. Thus the edge collision probability is low. However, BiGroup is more sensitive to the delay offset. The reason is that when the edges from two tags are too close to each other, it is hard to map each edge to each tag, resulting in errors in BiGroup.

XI. Conclusion

We have presented FlipTracer, a practical parallel decoding approach that is designed to work under highly dynamic backscatter system. FlipTracer achieves this by leveraging the stable transition pattern of the signals, rather than the dynamic and irregular IQ and time domain features of the signals. The experimental results show that FlipTracer is robust in various scenarios and achieves a nearly 2Mbps maximum throughput. In future works, we will explore the scalability of FlipTracer as well as hardware speedup for better time efficiency.

Appendix

We use a concrete example to explain why different assignments of the first layer clusters of OFG do not affect the decoding results. Figures 27(a) and 27(b) illustrate two different assignments of the Layer 1 clusters on the same OFG. In Figure 27(a), we denote Cluster 2 as HLL (that is to say, we denote the tag which causes the transition between cluster 1 and cluster 2 as Tag 1), Cluster 3 as LHL, and Cluster 4 as LLH. Assuming the reader receives Clusters 1, 2, and then 5, the output sequences are: Tag1: 'LHH'; Tag2: 'LLH'; Tag3: 'LLL'. If we use the assignments in Figure 27(b), the output sequences will be: Tag1: 'LLL'; Tag2: 'LHH'; Tag3: 'LLH'. However, the temporary IDs of the tags are just used for distinguishing these tags (but not used for denoting the tags) in the decoding process. Once we know there are three tags transmitting LLL, LLH, and LHH, respectively, such information is sufficient for correct decoding. After the whole packets of the three tags are decoded, we can get the IDs of the tags embedded in the packet.

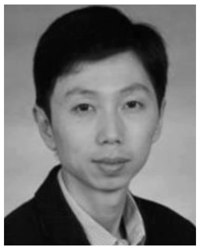
Acknowledgment

The authors would like to thank Prof. B. Li, Prof. B. Wu, Dr. Z. Wang, and Dr. L. Shangguan for their constructive support.

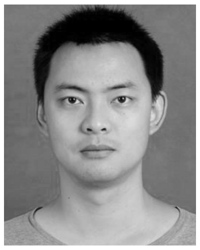
REFERENCES

[1] J. Wang and D. Katabi, "Dude, where's my card?: RFID positioning that works with multipath and non-line of sight," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 51–62, 2013.
[2] J. Xiong and K. Jamieson, "Arraytrack: A fine-grained indoor location system," in *Proc. NSDI*, 2013, pp. 71–84.
[3] L. Shangguan, Z. Li, Z. Yang, M. Li, and Y. Liu, "OTrack: Order tracking for luggage in mobile RFID systems," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 3066–3074.
[4] L. Yang, Q. Lin, X.-Y. Li, T. Liu, and Y. Liu, "See through walls with cots RFID system!" in *Proc. 21st Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2015, pp. 487–499.
[5] L. Yang, Y. Chen, X.-Y. Li, C. Xiao, M. Li, and Y. Liu, "Tagoram: Real-time tracking of mobile RFID tags to high precision using COTS devices," in *Proc. 20th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2014, pp. 237–248.
[6] L. Shangguan, Z. Yang, A. X. Liu, Z. Zhou, and Y. Liu, "Relative localization of RFID tags using spatial-temporal phase profiling," in *Proc. NSDI*, 2015, pp. 251–263.
[7] J. Ou, M. Li, and Y. Zheng, "Come and be served: Parallel decoding for cots RFID tags," in *Proc. 21st Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2015, pp. 500–511.
[8] P. Hu, P. Zhang, and D. Ganesan, "Leveraging interleaved signal edges for concurrent backscatter," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 18, no. 3, pp. 26–31, 2015.
[9] P. Hu, P. Zhang, and D. Ganesan, "Laissez-faire: Fully asymmetric backscatter communication," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 255–267, 2015.
[10] L. Kang, K. Wu, J. Zhang, H. Tan, and L. Ni, "DDC: A novel scheme to directly decode the collisions in UHF RFID systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 2, pp. 263–270, Feb. 2012.
[11] L. Kong, L. He, Y. Gu, M.-Y. Wu, and T. He, "A parallel identification protocol for RFID systems," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr./May 2014, pp. 154–162.
[12] C. Mutti and C. Floerkemeier, "CDMA-based RFID systems in dense scenarios: Concepts and challenges," in *Proc. IEEE Int. Conf. (RFID)*, Apr. 2008, pp. 215–222.
[13] A. N. Parks, A. Liu, S. Gollakota, and J. R. Smith, "Turbocharging ambient backscatter communication," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 619–630, 2015.
[14] J. Wang, H. Hassanieh, D. Katabi, and P. Indyk, "Efficient and reliable low-power backscatter networks," in *Proc. ACM SIGCOMM Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, 2012, pp. 61–72.
[15] D. Shen, G. Woo, D. P. Reed, A. B. Lippman, and J. Wang, "Efficient and reliable low-power backscatter networks," in *Proc. RFID*, 2009, pp. 61–72.
[16] C. Angerer, R. Langwieser, and M. Rupp, "RFID reader receivers for physical layer collision recovery," *IEEE Trans. Commun.*, vol. 58, no. 12, pp. 3526–3537, Dec. 2010.
[17] A. Bletsas, J. Kimionis, A. G. Dimitriou, and G. N. Karystinos, "Single-antenna coherent detection of collided FM0 RFID signals," *IEEE Trans. Commun.*, vol. 60, no. 3, pp. 756–766, Mar. 2012.
[18] R. S. Khasgiwale, R. U. Adyanthaya, and D. W. Engels, "Extracting information from tag collisions," in *Proc. IEEE Int. Conf. RFID*, Apr. 2009, pp. 131–138.
[19] S. Jana and S. K. Kasera, "On fast and accurate detection of unauthorized wireless access points using clock skews," *IEEE Trans. Mobile Comput.*, vol. 9, no. 3, pp. 449–462, Mar. 2010.
[20] D. Zanetti and B. Danev, "Physical-layer identification of UHF RFID tags," in *Proc. 6th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, Sep. 2010, pp. 353–364.
[21] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, Jun. 2014.
[22] *Radio-Frequency Identity Protocols Class-1 Generation-2*. Accessed: Jan. 2017. [Online]. Available: http://www.gs1.org/gsmp/kc/epcglobal/uhfc1g2
[23] D. K. Klair, K.-W. Chin, and R. Raad, "A survey and tutorial of RFID anti-collision protocols," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 3, pp. 400–421, 3rd Quart., 2010.
[24] M. Kodialam and T. Nandagopal, "Fast and reliable estimation schemes in RFID systems," in *Proc. 12th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, Sep. 2006, pp. 322–333.

Meng Jin received the B.S., M.S., and Ph.D. degrees in computer science from Northwest University, Xi'an, China, in 2012, 2015, and 2018, respectively. She is currently a Post-Doctoral Researcher with the School of Software and BNRist, Tsinghua University. Her main research interests include backscatter communication, wireless network co-existence at 2.4 GHz, mobile sensing, and clock synchronization.



Yuan He (SM'18) received the B.E. degree from the University of Science and Technology of China, the M.E. degree from the Institute of Software, Chinese Academy of Sciences, and the Ph.D. degree from The Hong Kong University of Science and Technology. He is currently an Associate Professor with the School of Software and BNRist, Tsinghua University. His research interests include Internet of Things, wireless and sensor networks, and mobile and ubiquitous computing. He is a member of the ACM.



Xin Meng received the B.S. degree in computer science from Northwest University, Xi'an, China, in 2012, where he is currently pursuing the master's degree. His main research interests include localization and wireless networks.

Yilun Zheng received the B.S. degree in electronic engineering from Tsinghua University in 2015, where she is currently pursuing the master's degree in software engineering. Her research interests are in the field of low-power wireless communication.

Dingyi Fang received the Ph.D. degree in computer application technology from Northwestern Polytechnical University, Xi'an, China, in 2001. He is currently a Professor with the School of Information Science and Technology, Northwest University, Xi'an. His current research interests include mobile computing and distributed computing systems, network and information security, and wireless sensor networks.

Xiaojiang Chen received the Ph.D. degree in computer software and theory from Northwest University, Xi'an, China, in 2010. He is currently a Professor with the School of Information Science and Technology, Northwest University. His current research interests include localization and performance issues in wireless ad hoc, mesh, and sensor networks, and named data networks.