FlipTracer: Practical Parallel Decoding for Backscatter Communication

Meng Jin
School of Information and Technology, Northwest University
mengj@stumail.nwu.edu.cn

Yuan He
School of Software and TNLIST, Tsinghua University
heyuan@tsinghua.edu.cn

Xin Meng
School of Information and Technology, Northwest University
mengxin@stumail.nwu.edu.cn

Yilun Zheng
School of Software and TNLIST, Tsinghua University
zhengyl15@mails.tsinghua.edu.cn

Dingyi Fang
School of Information and Technology, Northwest University
dyf@nwu.edu.cn

Xiaojiang Chen
School of Information and Technology, Northwest University
xjchen@nwu.edu.cn

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 Mbps, which is 6× higher than the state-of-the-art.

CCS CONCEPTS

• **Networks** → *Network architectures*; *Wireless access networks*;

KEYWORDS

Backscatter; Wireless; Parallel Transmission

MobiCom '17, October 16–20, 2017, Snowbird, UT, USA
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-4916-1/17/10…$15.00
https://doi.org/10.1145/3117811.3117828

1 INTRODUCTION

Backscatter provides the benefits of energy harvesting and low-power communication, making it attractive to a broad class of applications [16, 17, 20–23]. These applications, such as warehouse inventories, object tracking, and industry surveillance, involve large volumes of data carried by a large number of deployed tags. 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 network coding, but it incurs computational overhead on the tags, which may not be realistic. More recent works [6, 7, 13] 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.

In order 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' 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 constructing 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 [13].

In the rest of the paper, we discuss related work in Section 2, and describe the motivation of our design in Section 3. The design details of FlipTracer are introduced in Sections 5∼7. Sections 8 and 9 evaluate the performance analytically and empirically. We conclude the paper in Section 10.

2 RELATED WORK

There have been many efforts made to enable multiple accesses in backscatter communication. The existing commodity backscatter systems typically adopt the slotted aloha scheme [1]. This scheme however incurs high coordination overhead and low throughput, especially when collision occurs. Although some recent methods [2] can deal with the collision problem, the throughput is bounded at one packet per time slot. To further improve the throughput, recent works try to parallelize multiple backscatter transmissions:

Using coding mechanisms. A direct method for parallel transmission is to exploit coding mechanisms [5, 9, 11, 12, 14] on tags to facilitate collision recovery. In these methods, each tag exploits an orthogonal code for encoding data where each bit is converted to 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 [3, 4, 6, 7, 10, 13, 18, 19]. Specifically, in these works, the reader decode the collided signals according to the signals' features in time and/or IQ domain:

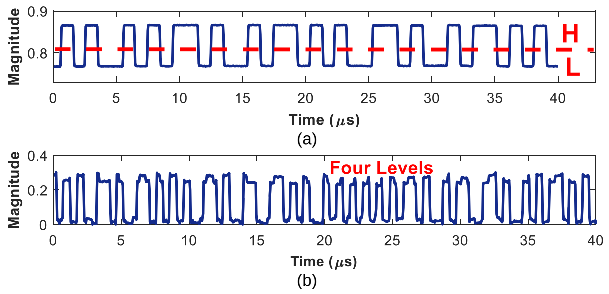


Figure 1: Signals in time domain: (a) a single tag. (b) two tags.

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 [7, 18, 19] propose to identify the collided signals based on their IQ domain positions.

Using tags' time domain features. An alternative method that has been considered in prior works is separation in the time domain [7, 13]. 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, and connect with the signal clusters in the IQ domain for collision decoding. For example, LF-Backscatter [7] applies folding to extract the periodical signal edges of individual tags. BiGroup [13] 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. In practice, however, signals from the tags never exhibit stable patterns. According to our experiment, the dynamics induced by the low-cost hardware components and the ambient environment severely undermines signal features in both the time and the IQ domains. 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.

3 BACKGROUND AND MOTIVATION

3.1 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,

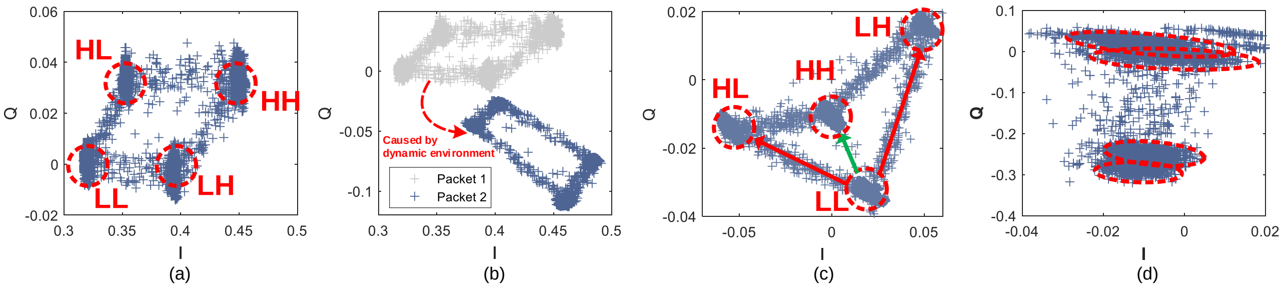


Figure 2: Symbol clusters: (a) a 2-tag collision example. (b) collided signal from two sequential packets under dynamic working condition. (c) 2-tag collision with non-linear dependency. (d) overlapped clusters.

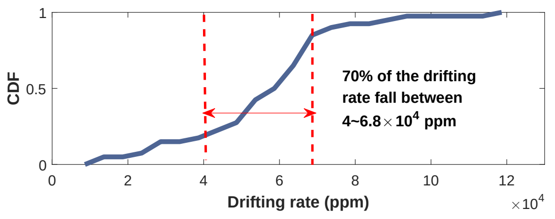


Figure 3: CDF of the measured drifting rate of 20 tags.

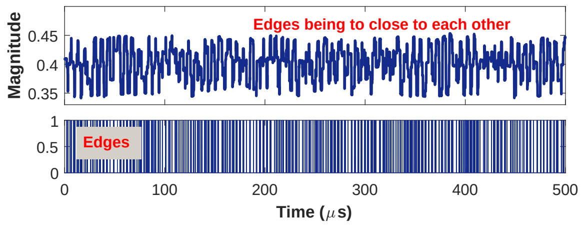


Figure 4: Signal series of a 4-tag collision.

denoted as S = $[S_1, S_2, ..., S_N]$, 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. Therefore, 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 $i = 1, ..., N$. Then the collision is decoded. However, this is a challenging task due to the dynamics in the backscatter signals.

3.2 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.

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 [19] 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 [13]. 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.

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 [7]). However, due to the low cost design, the oscillators used in backscatter tags typically exhibit high drifting rates [7, 8, 24]. 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 ∼ 68μs time offsets within one-packet duration. Since the bit duration is only 10μs, 40 ∼ 68μs offsets will inevitably incur excessive errors in the edge identification process.

Some schemes [13] propose to address this problem by using linear regression. Such schemes work only if the signal 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.

3.3 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

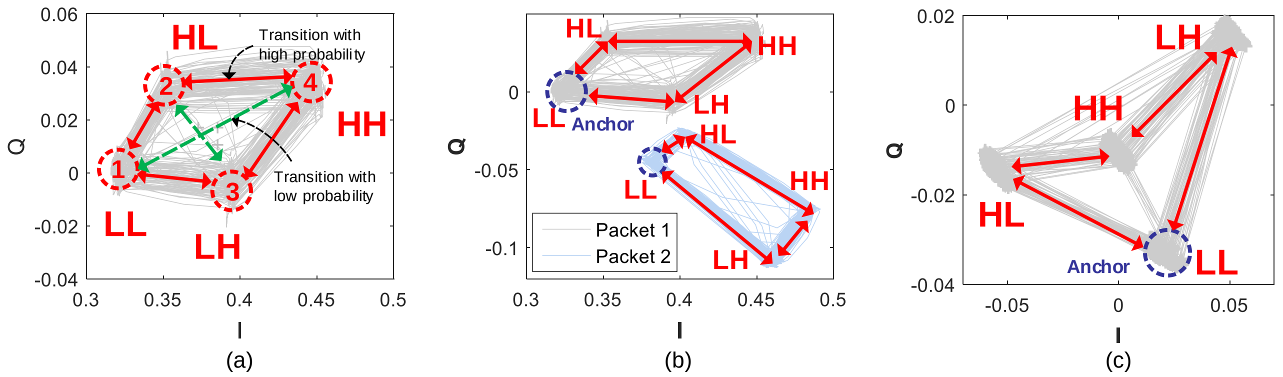


Figure 5: Transition pattern among clusters: (a) a 2-tag collision example. (b) collision under a dynamic working condition. (c) collision with non-linear dependency

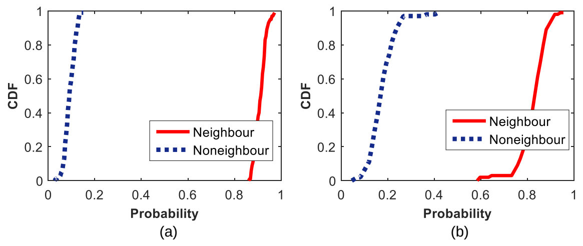


Figure 6: The CDF for the transition probability between both neighbor and non-neighbor clusters: (a) two tags; (b) five tags.

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., "L_1L_2" and "L_1H_2"), while the concurrent flipping of multiple tags causes the transition between two clusters whose combined states have multiple different states (e.g., "L_1L_2" and "H_1H_2"). We term the former cluster pair 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, 13], the transition probabilities between the neighbor clusters are obviously higher than those between the non-neighbor clusters.

As a concrete example, consider the transition probability between non-neighbor clusters in a 2-tag collision case where each tag transmits at 100Kbps and the USRP reader samples at 25 MHz (i.e. the reader samples 250 times for each bit duration). 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: $1-(83\cdot82)/(83\cdot83) = 1.2\%$. Moreover, even under high concurrency scenarios, we can still observe very low transition probability between non-neighbor clusters (detailed discussion is given in Section 8.2).

We further conduct experiments to show the *distinctness* and *consistency* of the transition probability between neighbor clusters.

Distinctness. We plot the CDFs of the transition probabilities between both neighbor and non-neighbor clusters in Figure 6(a), 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× higher than that between non-neighbor clusters. This ensures that we can correctly find the neighbors for each cluster.

Consistency. We have shown in Figure 5 that the signal's transition pattern is stable under dynamic scenarios, now let us look at its stability when the concurrency level is high. Figure 6(b) plots the CDFs of the transition probabilities between neighbor and non-neighbor clusters in a 5-tag collision case. We can see that when the concurrency level is high, the transition probabilities between neighbor and non-neighbor clusters are still distinguishable.

4 FLIPTRACER OVERVIEW

FlipTracer is a practical parallel decoding approach in which the collided signals are identified based on the transition probabilities between symbol clusters. In this system, the tags transmit packets concurrently when they receive a query from the reader. The collided packets will be resolved on the reader-side after every transmission. The FlipTracer system architecture is shown in Figure 7:

- **OFG construction.** FlipTracer first clusters the physical symbols in the IQ plane. Then, based on the observed signal transition probabilities among symbol 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 the cluster identification component.
- **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.

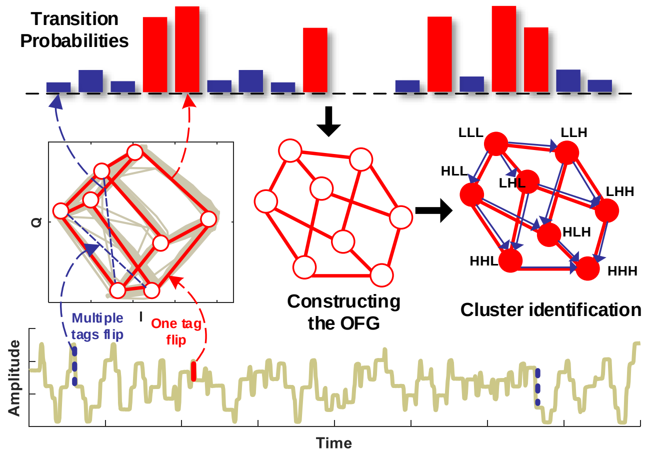


Figure 7: Workflow of FlipTracer.

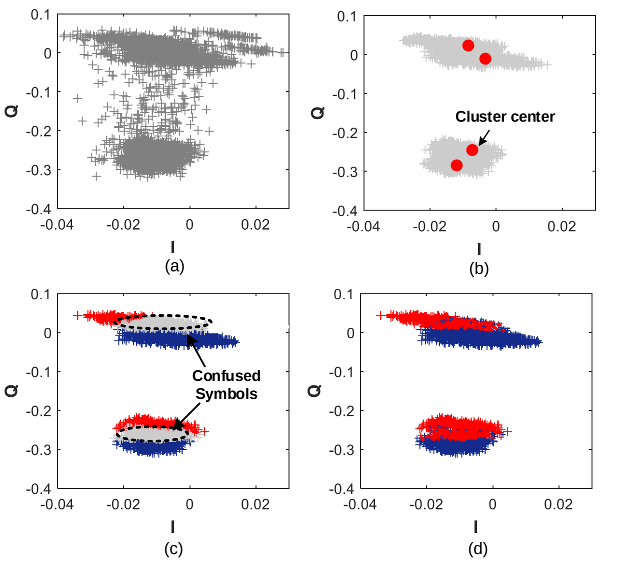


Figure 8: Symbol clustering: (a) received symbols. (b) identified cluster centers. (c) confused symbols. (d) symbols are finally classified to each cluster.

5 THE ONE-FLIP-GRAPH

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

5.1 Symbol Clustering

The first challenge we face is reliable symbol clustering. Although symbol clustering methods have been proposed in some recent works [7, 13], they can not separate overlapping clusters which are common in noisy environments. To separate the overlapping clusters, we need to first find their centers. Hence we employ *local densities*[1] based clustering methods. Specifically, a cluster center typically has high local density and is surrounded by symbols with lower local densities. Although clusters overlap with each other, we can still find their centers by finding the density peaks.

Our clustering approach is designed based on the algorithm introduced in [15] (we call it LDBC in this paper). We first use the idea in LDBC to find the Nc cluster centers, and estimate for each symbol i the probability that it belongs to each cluster C_k ($C_k \in$ **C** $= \{C_1, ..., C_{Nc}\}$) as **Pd**$(i) = \{pd_i^k \mid 1 \le k \le Nc\}$, where pd_i^k is determined by the location and the local density of symbol i.

Then it seems that we can directly classify symbol i to C_k if pd_i^k is the maximum in **Pd**(i). However, since the symbols located at the overlapping area (termed as *confused symbols*) may exhibit similar pd_i^k for the corresponding clusters, classifying the confused symbols based only on their **Pd**(i) 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 for a while between two transitions, thus successive symbols are likely to belong to the same cluster. We propose a parameter **Pt**$(i) = \{pt_i^k \mid 1 \le k \le Nc\}$ to describe from time domain that how likely symbol i belongs to cluster C_k. We have $pt_i^k = \frac{\#C_k}{w_c}$, where $\#C_k$ denotes the number of symbols belonging to cluster k in the time window centered at symbol i. The length of the window is set according to the bit duration of the tags and the sampling rate of the reader.

[1]The local density of a symbol is defined as the number of symbols located within a cutoff distance.

Now we can classify the confused symbols based on a joint consideration of **Pd** and **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 $\mathbf{P_{cluster}}(i)$.

FlipTracer goes through three steps to classify the symbols (Figure 8 gives an example of the symbol clustering process for a 2-tag collision):

- It uses the idea in LDBC to estimate the **Pd** for each symbol (as shown in Figure 8(b)).
- It classifies each symbol i to the cluster which achieves $pd_i^k > PD_{th}$ (we empirically set $PD_{th} = 0.4$). If there are multiple (or no) clusters achieve $pt_i^k > PD_{th}$, it denotes symbol i as a confused symbol (as shown in Figure 8(c)).
- It estimates $\mathbf{P_{cluster}}$ for the confused symbols, and classifies the confused symbols based on $\mathbf{P_{cluster}}$. The final clustering result is shown in Figure 8(d).

Now each symbol is marked by the label of its cluster.

5.2 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 3, 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.

However, we face a problem that due to the reader's query, signal edges of different tags might be aligned during the initial period of a concurrent transmission (as shown in Figure 9(d)). It results in high transition probabilities between non-neighbor clusters, in the initial period (as shown in Figure 9(a)). If such alignment lasts

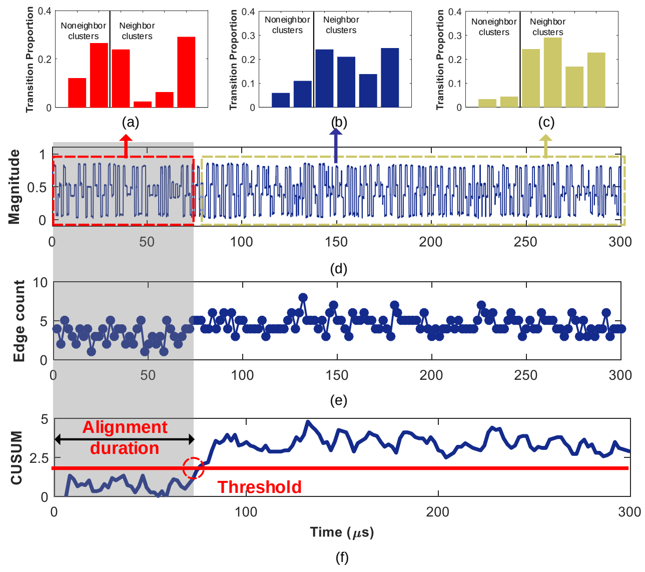


Figure 9: Impact of aligned signal flipping on the transition probabilities: (a)-(c) are the transition probabilities between clusters during the alignment period, the entire signal series and the misalignment period, respectively. (d) signal series. (e) the number of the state flipping. (f) CUSUM of the state flipping number.

for a long period of time (which is caused by very similar clock rates between tags), the gaps between the transition probabilities for neighbor and non-neighbor clusters will be small. This may incur errors in the neighbor identification process. Therefore, for reliable neighbor identification, we would like to first filter out those aligned signals, then calculate the transition probabilities based on only the misaligned signals.

Filter out the aligned signal flipping. The intuition of our method for detecting the aligned signals is that state flipping occurs more frequently in the misaligned signals than in the aligned signals. Figure 9(e) shows the number of state flipping counted over successive windows. Obviously, signal alignment leads to less frequent signal flipping. However, we cannot find an appropriate threshold to separate the aligned and misaligned signals.

To solve this problem, we use the Cumulative Sum (CUSUM) test, a method designed to detect the change in data distribution, to detect the change in flipping frequency. We apply the CUSUM on top of the series in Figure 9(e), and show the cumulative sum in Figure 9(f). We can see that now the aligned and misaligned signals are separable. After filtering out the aligned signals, the transition probabilities between neighbor clusters are highly distinct from those between non-neighbor clusters (as shown in Figure 9(c)).

Finding the neighbors. 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:

$$P_{trans}(C_i, C_j) = \frac{\#(C_i \leftrightarrow C_j)}{\sum_{C_k \in \mathbf{C}} \#(C_i \leftrightarrow C_k)} \tag{1}$$

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 $\mathbf{C}$. Transitions between clusters can be detected using the method given in [13].

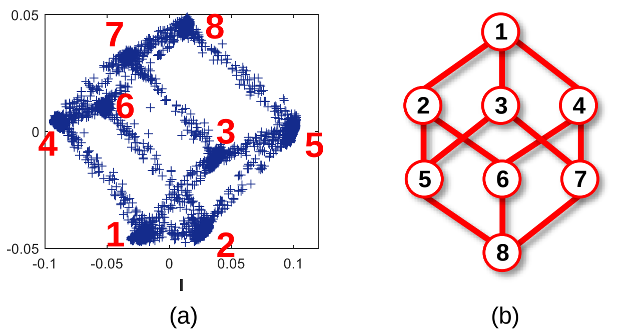


Figure 10: OFG example: (a) received symbol samples of a 3-tag collision. (b) constructed OFG for the received symbol clusters.

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:

$$Conf(C_i) = \frac{min\{P_{trans}(C_i, \mathbf{C_{pos}}(C_i))\}}{max\{P_{trans}(C_i, \mathbf{C_{nopos}}(C_i))\}} \tag{2}$$

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.

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}.

5.3 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. As an example, Figure 10(a) shows the received symbols of a 3-tag collision case, and Figure 10(b) shows the corresponding OFG. One problem we face, however, 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.* Specifically, a loop with an odd number of nodes implies that for an arbitrary cluster on the loop, its combined

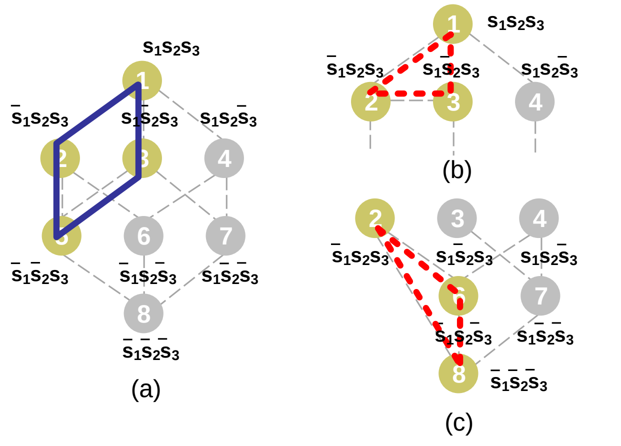


Figure 11: Loops in the OFG of a 3-tag collision case: (a) an example of a correct loop. (b) and (c) are two examples of the abnormal loops.

state can transfer to the initial state through an odd number of transitions, which is however impossible. The reason is that for a combined state $[S_1, ..., S_N]$, its each state S_i has to experience two transitions to transfer to its initial state (i.e., $S_i \to \overline{S}_i \to S_i$). Thus the combined state $[S_1, ..., S_N]$ has to experience an even number of transitions to transfer to the initial combined state. Figure 11(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. 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 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 11(b) and 11(c) show two examples of such abnormal loops.

The above observation indicates that we can avoid forming false connections by detecting such *abnormal loops*. Specifically, before forming a connection $< C_i, C_j >$, we will first check whether $< C_i, C_j >$ leads to an abnormal loop. If it does, we will label C_j as a non-neighbor cluster of C_i [2].

6 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 when all tags are being charged.

To describe the cluster identification process more clearly, we have to first layer the OFG. Specifically, the anchor cluster is the root (Layer 0) 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.

Now we can identify the clusters in the OFG layer by layer, starting from C_{root}.

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

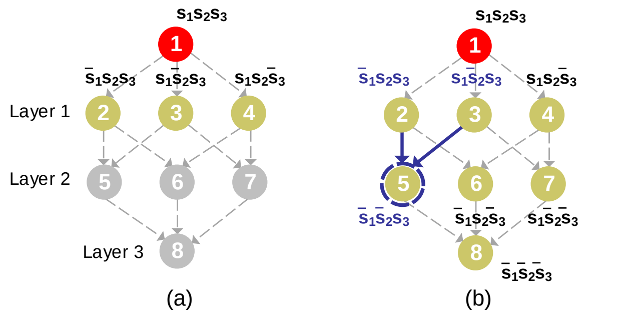


Figure 12: The process of decoding a 3-tag collision: (a) Identify the first layer clusters. (b) Identify the clusters in Layer $l - 1$.

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 $[S_1, S_2, ..., S_N]$, we can identify its neighbor clusters (i.e., the Layer 1 clusters) as $[\overline{S}_1, S_2, ..., S_N]$, $[S_1, \overline{S}_2, ..., S_N]$, ..., $[S_1, S_2, ..., \overline{S}_N]$. Figure 12(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 show in the appendix with a concrete example that different assignments of the Layer 1 nodes do not affect the decoding result.

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 12 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 $S_{k_1}, S_{k_2}, ..., S_{k_l}$, where $k_1, k_2, ..., k_l$ 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 $\{[S_{k_2}, S_{k_3}, ..., S_{k_{l-1}}, S_{k_l}], [S_{k_1}, S_{k_3}, ..., S_{k_{l-1}}, S_{k_l}], ..., [S_{k_1}, S_{k_2}, ..., S_{k_{l-2}}, S_{k_{l-1}}]\}$. Thus if the combined states of

[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.

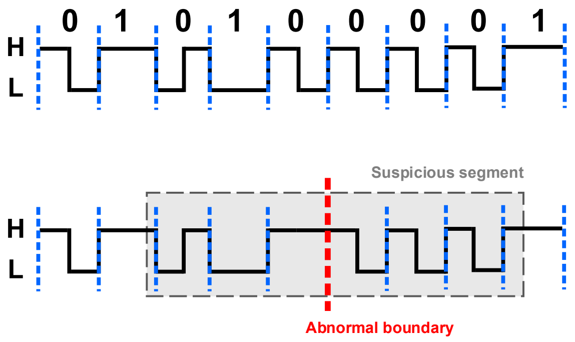


Figure 13: FM0 coding: (a) A correct state sequence. (b) A state sequence with errors.

the Layer-$(l-1)$ clusters have been identified (which means that $k_1, k_2, ..., k_l$ are known), the combined state of C_l^i can be identified accordingly. As the example shown in Figure 12(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.

The process of cluster identification can be concluded as follow:

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

Then the combined states of all the clusters can be identified.

7 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. For brevity, we will later use FM0 code as a vehicle to describe our error correction method. Our method can be generalized to handle Miller codes as well with slight modifications.

FM0 code inevitably inverts the signal state at every bit boundary (as shown in Figure 13(a)). Thus it is a obvious error if two chips[4] 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 13(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 $k = 1, 2, ..., K$ 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 $\{S_1, S_2, ..., S_K\}$ to all the chips $k = 1, 2, ..., K$ is given as follows:

$$P(\mathrm{S}) = \beta \prod_{k} \phi(S_k) \prod_{k,k+1} \psi(S_k, S_{k+1}) \tag{3}$$

where β serves as a normalization constant. The factor $\phi(S_k)$ represents the emission probability of assigning S_k to k, which is highly related to the confidence of the clustering process (i.e., $\mathbf{P_{cluster}}$). Thus $\phi(S_k)$ can be given by the $\mathbf{P_{cluster}}$ of the symbols in $\mathbf{C_{S_k}}$, where $\mathbf{C_{S_k}}$ represents the clusters indicating a S_k state of tag i. The factor $\psi(S_k, S_{k+1})$ in equation (3) presents 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 $\psi(S_k, S_{k+1}) = 1$ if $S_{k+1} = S_k$, and $\psi(S_k, S_{k+1}) = 0$ if $S_{k+1} = \overline{S}_k$. Otherwise, we have $\psi(S_k, S_{k+1}) = 0.5$.

We calculate $P(\mathrm{S})$ for all possible state assignments and identify the one which the maximum $P(\mathrm{S})$ as the correct assignment for the corresponding sequence.

8 ANALYSIS

In this section, we provide theoretical analysis on the performance of FlipTracer.

8.1 Computation Overhead and Latency

The computation overhead of FlipTracer is dominated by the symbol 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 (which are much fewer than symbols) and cluster those grids (similar with BiGroup [13]). This reduce the computation overhead to $O(k \cdot logk + M)$, where k is the number of grids and M is the number of confused samples. In FlipTracer, k is around 200 and M is influenced by the SNR of the collided signals (the average number is 80 according to our experiments).

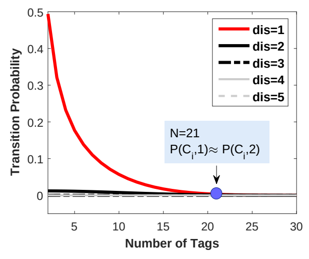
Let N be the number of the tags. The computation overheads of the OFG construction and cluster identification components are $O(N^2 + 2^{N-1} \cdot N)$ and $O(2^{N-1} \cdot N)$, respectively. Since the sampling capacity of the reader is limited, the aggregated throughput cannot increase infinitely with the number of tags. We find through experiments that the performance of FlipTracer peaks at $N = 5$.

We implement FlipTracer on an USRP N210 connected to a general PC equipped with 3.6GHz CPU and 16G memory. FlipTracer takes 180∼320μs to decode the collided packet when $N = 5$. Our theoretical analysis tells that the upper bound of aggregated throughput that FlipTracer can achieve is "8 tags transmit at 500Kbps". In this case, we can still keep the time overhead in *ms* level.

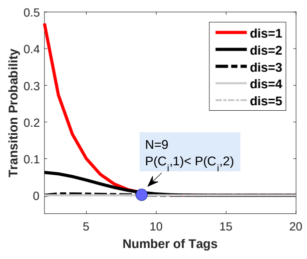
8.2 Upper Bound Throughput

The aggregated throughput of FlipTracer is determined by two factors: the number of tags and the bitrate of each individual tag. Since the sampling capacity of the reader is limited, when the number of tags scales or the tags transmit at higher bitrates, signal edges of different tags may be aligned with each other with a higher probability. When the proportion of the aligned edges reaches a certain level, FlipTrace is likely to build false connections when constructing the OFG. This leads to errors in the decoding process.

[4]The FM0 code uses two chips to represent one bit



(a) Bitrate=100Kbps



(b) Bitrate=500Kbps

Figure 14: Transition probability vs. number of tags

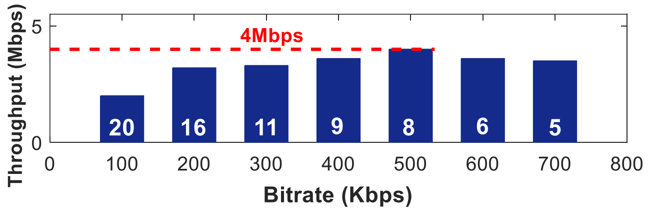


Figure 15: Maximum throughput vs. bitrates.

As noted earlier, if the transition probabilities between a cluster 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. Assume that N tags transmit concurrently with the same bitrate B and the sampling rate of the reader is S. Further assume that the signal edge has a width of e samples at the reader's sampling rate. The transition probability between C_i and another cluster C_j, can be calculated as follows, where $d(C_i, C_j) = dis$[5]:

$$P(C_i, dis) = \frac{\binom{\frac{S}{B \cdot e}}{1} \cdot \binom{\frac{S}{B \cdot e}-1}{N-dis} \cdot (N-dis)!}{\frac{S}{B \cdot e}^N} \tag{4}$$

If we have $P(C_i, 1) > P(C_i, dis)$ and $dis \geq 2$ for each C_i, we can correctly construct the OFG.

Figures 14(a) and 14(b) show the transition probability vs. the number of concurrent tags (under different dis), with the bitrates of the tags set at 100Kbps and 500Kbps, respectively. We find in Figure 14(a) that when $N \geq 21$, the transition probabilities between neighbor and non-neighbor clusters are too close to be differentiated. In this case ($B = 100Kbps$), the maximum throughput is 2.4Mbps. Similarly, Figure 14(b) shows that the maximum throughput is 4Mbps for $B = 500Kbps$. We also find that the transition probabilities between non-neighbor clusters decrease as N increases. This is because the number of clusters increases exponentially with that of the tags. Therefore, the transition probability between each two clusters will decrease accordingly.

We further show the maximum throughput of FlipTracer under different bitrates in Figure 15. Figure 15 presents the theoretical upper bound of the throughput that FlipTracer can achieve, i.e., 4Mbps (concurrent transmissions from 8 tags at 500Kbps)

[5]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.

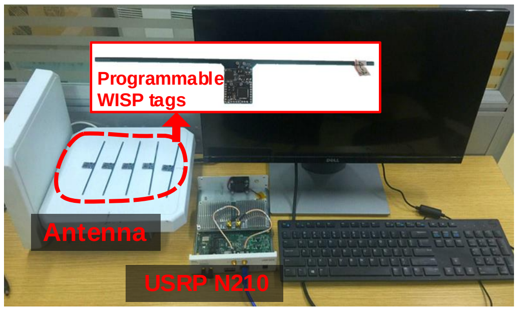


Figure 16: Experimental setup.

9 EVALUATION

9.1 Experiment Setting

The hardware platform is shown in Figure 16.

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 as 20MHz, and the Tx-Gain is 20dBm.

Backscatter tags. Since COTS RFID manufacturers only provide limited interfaces to developers, we implement FlipTracer on programmable WISP tags. This only requires a slight modification to the EPCglobal C1G2 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.

We compare FlipTracer with the following three 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 [13] 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.

9.2 Evaluation of OFG construction

We first focus on the OFG construction component. As a core component of FlipTracer, it has a significant impact on the overall performance.

Two factors can affect the performance of this component - the number of concurrent tags and the SNR. In practice, a higher concurrency means a higher probability for multiple tags to flip their states concurrently, which may lead to higher error rate when identifying neighbour clusters. On the other hand, a lower SNR will incur more errors in both the symbol clustering and the neighbour identification processes.

In the experiments, we increase the number of tags from 2 to 5. For each concurrency level, we change the SNR of the signals by adjusting the gain of the transmitting antenna from 0 to 20dBm. We collect 100 collided packets for each setting and show the corresponding error ratio of the OFG construction component in Figure 17. The error ratio is calculated as the number of wrong OFGs over the constructed 100 OFGs. We find that: i) when the number of

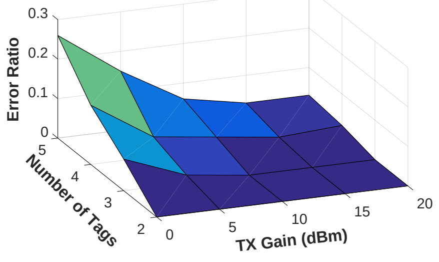


Figure 17: Error ratio of the OFG construction component.

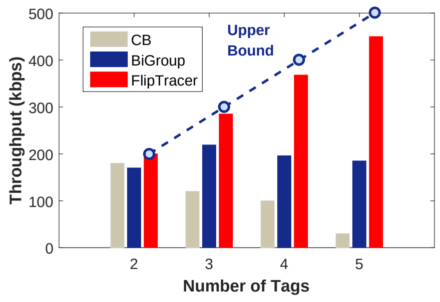


Figure 18: Aggregated throughput with different numbers of tags.

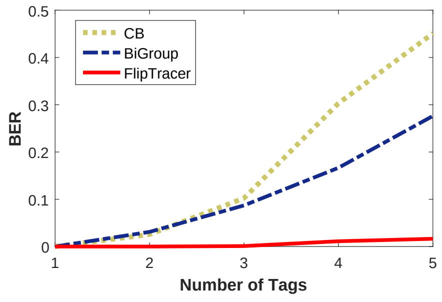


Figure 19: BERs with different numbers of colliding tags.

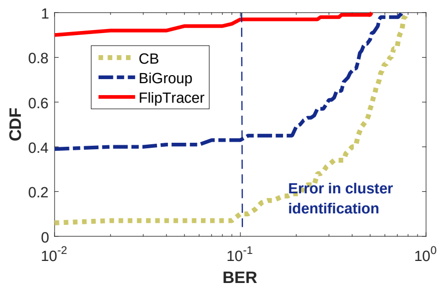


Figure 20: CDFs of BERs for different schemes when the number of tag is 5.

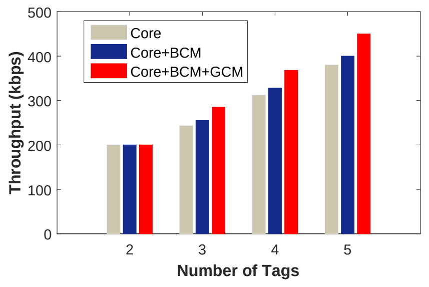


Figure 21: Breakdown of each component's contribution to throughput.

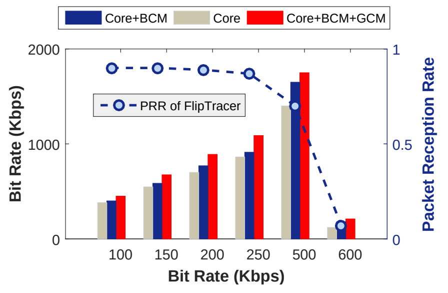


Figure 22: Throughput under different bitrates.

tags ≤ 4, the error ratio can be controlled under 5% even when the TX Gain is 0dBm; and ii) when the TX Gain ≥ 10dBm, the error ratio can be controlled under 10% (the TX Gain of the commercial RFID reader is 30dBm). Clearly, FlipTracer can correctly construct an OFG with high reliability.

9.3 FlipTracer Goodput

In this section, we look at the benefit of FlipTracer in terms of throughput and BER.

Aggregated throughput. Figure 18 compares the aggregated throughput achieved by different schemes for different numbers of colliding 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.

BER. Figure 19 plots the BERs of different schemes. When collision occurs, the decoding scheme may output multiple packets. Average BERs are record by comparing each output packet and the transmitted counterpart.

In Figure 19, 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 if the number of tags is larger than 4. This is because the error probability for cluster identification increases 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 20 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.

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 21 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: the core design of FlipTracer does really well by itself, but there are more errors as the number of tags increases. 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.

9.4 Impacts of Practical Factors

To understand the impacts of some practical factors on FlipTracer, we conduct the following experiments.

Bitrate. As shown in Section 8, besides the concurrency level, bitrate is another factor that can affect the performance of FlipTracer. In this experiment, we vary the bitrate of the 5 tags from 100Kbps to 600Kbps, and observe the aggregated throughput. Since the WISP platforms currently support only a 256Kbps bitrate, we conduct simulations to see the performance of FlipTracer with 500∼600 Kbps

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

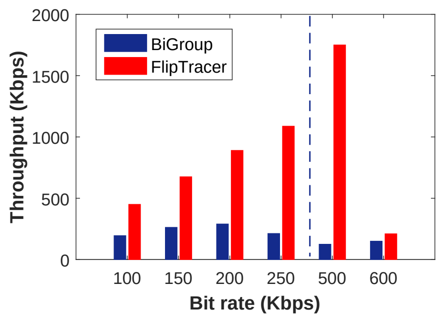


Figure 23: Throughput comparison under different bitrates.

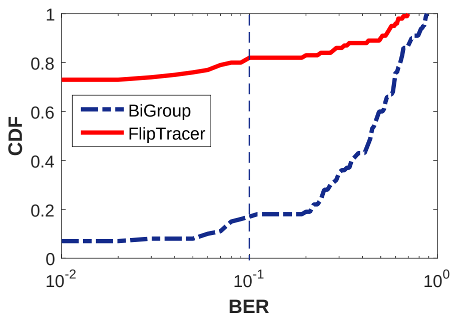


Figure 24: BERs when bitrate is 500Kbps and the number of tag is 5.

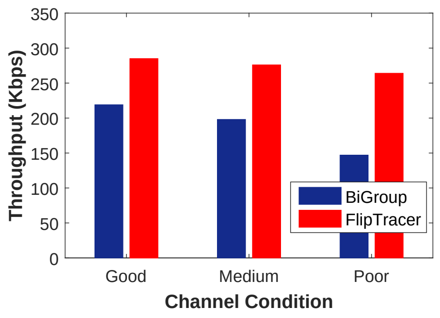


Figure 25: Throughput comparison under different channel condition.

bitrate. Figure 22 shows that the aggregated throughput crash at 600Kbps. This gives us an empirical upper bound on the aggregated throughput FlipTracer can support - **concurrent transfer from 5 tags at 500Kbps**. The crash is not surprising. Our reader samples at 20 MHz, and tags transmit at 600Kbps, so we can stack at most 12 separated edges in one chip (half bit) duration. Our deployment have 5 nodes, so there is a large number of aligned signal edges, which may overthrow our assumption that "the signal edges from different tags are separated most of the time".

In Figure 23, We compare the aggregated throughput for 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 crash 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 for each tag is not stable, the reader can hardly distinguish the edges of different tags. This results in excessive errors in the decoding process of BiGroup. Figure 23 also shows that the maximum aggregated 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× higher than that of BiGroup.

In Figure 24, 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 20 and 24, 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.

Channel condition. Channel condition can also affect the performance of FlipTracer. To evaluate the impact of channel condition, in this experiment, we use three tags (each transmit at 100Kpbs) and keep moving the tags further and further away from the reader to worsen the channels of all tags. Figure 25 compares the aggregated throughput of BiGroup and FlipTracer under different channel conditions. We can see that when the channel quality is good, both FlipTracer and BiGroup achieve good throughput. When the channel worsens, the aggregated throughput of BiGroup decreases significantly. The reason behind is that under a poor channel condition, clusters become closer to or even overlap with each other, which brings errors in symbol clustering process of BiGroup. What's worse, noise spike occurs frequently in poor channel conditions, which will be confused as signal edges, making it more difficult to separate and extract the signal edges of different tags. This incurs excessive errors in BiGroup. By contrast, FlipTracer is highly robust to poor channel conditions.

We further plot the CDFs of BERs for BiGroup and FlipTracer under the poor channel condition in Figure 26. We find that the channel condition can affect the accuracy in both symbol clustering and cluster identification components, leading to high BERs even when the number of the concurrent tags is 3.

9.5 FlipTracer in "bad" working condition

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

Dynamic environment. Figure 27 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 and the reader; 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 27 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 the tags, FlipTracer is able to decode the collided signals without channel coefficient information.

Co-existence of fast and slow tags. One of the key benefits of FlipTracer is that it can support widely different bitrates among tags. To evaluate this benefit, we use three tags (denoted as Tags 1, 2, and 3) 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 different bitrates (i.e., 100Kbps, 150Kbps and 200Kbps). iii) we let Tags 1 and 2 transmit very slow (50Kbps) and Tag 3 transmit very fast (250Kbps).

Figure 28 shows the packet reception rate achieved by each tag under different scenarios. The results show that: i) the throughput achieved in the second case is almost the same as that achieved in the first case. ii) even in the third case where the bitrate of the fast tag is 5× as that of the slow tags, the throughput only slightly decreases. The above results imply that FlipTracer can support concurrency tags with widely different bitrates.

9.6 Impact of unstable clocks

Another key benefit of FlipTracer is its ability to decoding without stable clocks. Since clock drifting rates of the tags are uncontrollable

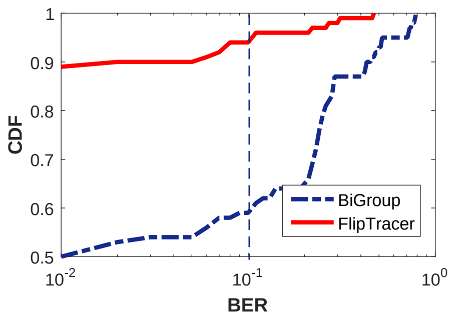


Figure 26: CDF for BERs under poor channel condition.

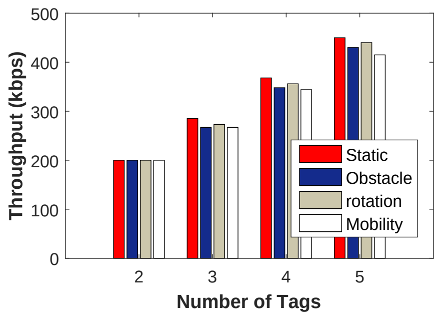


Figure 27: Throughput of FlipTracer under different working conditions.

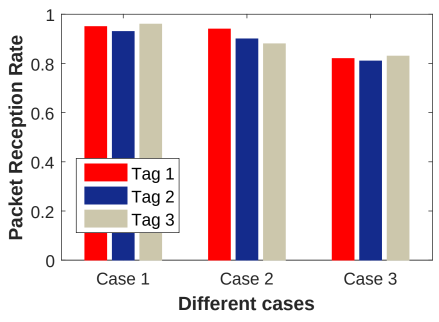


Figure 28: Throughput of FlipTracer when tags transmit at different bitrates.

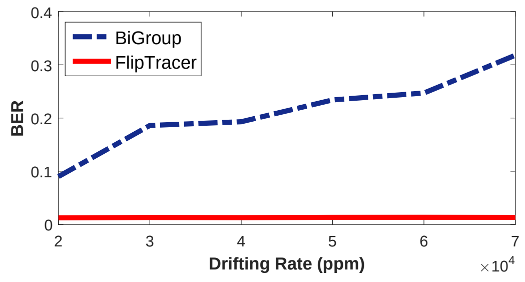


Figure 29: Impact of clock drifting rate.

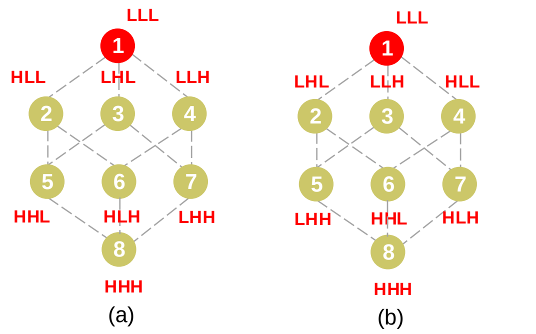


Figure 30: Two different assignments of the Layer 1 clusters on the same OFG.

in the real-world experiment, we conduct simulations to evaluate the impact of the clock drifting rate. Our simulation inherits the settings from our experiments.

Figure 29 shows the BER comparison of BiGroup and FlipTracer under different clock drifting rates, when the number of colliding tags is 5. We can see that the BER of BiGroup increases rapidly when the drifting rate increases. In comparison, the increase in the BER of FlipTracer is almost imperceptible.

10 CONCLUSIONS

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 aggregate throughput. In the future works, we would like to explore the scalability of FlipTracer as well as hardware speedup for better time efficiency.

ACKNOWLEDGMENT

We would like to thank the shepherd Prof. Prasun Sinha, as well as Prof. Baochun Li, Prof. Bin Wu, Prof. Zheng Wang, Prof. Longfei Shangguan, and the anonymous reviewers for their valuable feedbacks on this paper. This work was supported by National Basic Research Program (973 program) under Grant of 2014CB347800, National Natural Science Fund of China for Excellent Young Scientist No.61422207, the NSFC (61572402, 61672428, 61672427), and the Program for Innovative Research Team in University No. IRT13090.

A APPENDIX

We explain why different assignments of the first layer clusters of OFG do not affect the decoding results. In fact, the assignments only determine the "temporary IDs" of the tags. All the assignments will eventually output N *uniquely determined* 'HL' sequences (where N is the number of tags). In other words, different assignments only lead to different mappings between the N "temporary IDs" and the N sequences. Note that what we essentially care about is not the "temporary IDs" (which are used to distinguish different tags in the decoding process), but the IDs (or data) embedded in the packets (these IDs are physically associated the tags).

Here we use a concrete example to clearly explain the above point. Figures 30(a) and 30(b) illustrate two different assignments of the Layer 1 clusters on the same OFG. In Figure 30(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 0, 1, and then 5, the output sequences are: Tag1: 'LHH'; Tag2: 'LLH'; Tag3: 'LLL'. If we use the assignments in Figure 30(b), the output sequences will be: Tag1: 'LLL'; Tag2: 'LHH'; Tag3: 'LLH'. As noted above, 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 know the IDs (or data which are embedded in the payloads of the packets) of the tags.

REFERENCES

[1] Radio-frequency identity protocols class-1 generation-2. http://www.gs1.org/gsmp/kc/epcglobal/uhfc1g2.

[2] O. Abari, D. Vasisht, D. Katabi, and A. Chandrakasan. Caraoke: An e-toll transponder network for smart cities. In *SigComm*, 2015.

[3] C. Angerer, R. Langwieser, and M. Rupp. Rfid reader receivers for physical layer collision recovery. *IEEE Transactions on Communications*, 58(12):3526–3537, 2010.

[4] A. Bletsas, J. Kimionis, A. G. Dimitriou, and G. N. Karystinos. Single-antenna coherent detection of collided fm0 rfid signals. *IEEE Transactions on Communications*, 60(3):756–766, 2012.

[5] A. Gudipati and S. Katti. Strider: Automatic rate adaptation and collision handling. In *SIGCOMM*, 2011.

[6] P. Hu, P. Zhang, and D. Ganesan. Leveraging interleaved sig-nal edges for concurrent backscatter. In *HotWireless*, 2014.

[7] P. Hu, P. Zhang, and D. Ganesan. Laissez-faire: Fully asymmetric backscatter communication. In *SIGCOMM*, 2015.

[8] S. Jana and S. K. Kasera. On fast and accurate detection of unauthorized wireless access points using clock skews. In *MobiCom*, 2008.

[9] L. Kang, K. Wu, J. Zhang, and L. M. Ni. Ddc: A novel scheme to directly decode the collisions in uhf rfid systems. *IEEE Transactions on Parallel and Distributed Systems*, 23(2):263–270, 2012.

[10] R. Khasgiwale, R. Adyanthaya, and D. Engels. Extracting information from tag collisions. In *RFID*, 2009.

[11] L. Kong, L. He, Y. Gu, M.-Y. Wu, and T. He. A parallel identification protocol for rfid systems. In *INFOCOM*, 2014.

[12] C. Mutti and C. Floerkemeier. Cdma-based rfid systems in dense scenarios: Concepts and challenges. In *RFID*, 2008.

[13] J. Ou, M. Li, and Y. Zheng. Come and be served: Parallel decoding for cots rfid tags. In *MobiCom*, 2015.

[14] A. N. Parks, A. Liu, S. Gollakota, and J. R. Smith. Turbocharging ambient backscatter communication. In *SIGCOMM*, 2014.

[15] A. Rodriguez and A. Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, 2014.

[16] L. Shangguan, Z. Li, Z. Yang, M. Li, and Y. Liu. Otrack: Order tracking for luggage in mobile rfid systems. In *INFOCOM*, 2013.

[17] L. Shangguan, Z. Yang, A. Liu, Z. Zhou, and Y. Liu. Relative localization of rfid tags using spatial-temporal phase profiling. In *NSDI*, 2015.

[18] D. Shen, G. Woo, D. P. Reed, A. B. Lippman, and J. Wang. Efficient and reliable low-power backscatter networks. In *RFID*, 2009.

[19] J. Wang, H. Hassanieh, D. Katabi, and P. Indyk. Efficient and reliable low-power backscatter networks. In *SIGCOMM*, 2012.

[20] J. Wang and D. Katabi. Dude, where's my card? rfid positioning that works with multipathand non-line of sight. In *SIGCOMM*, 2013.

[21] J. Xiong and K. Jamieson. Arraytrack: A fine-grained indoor location system. In *NSDI*, 2013.

[22] 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 *MobiCom*, 2014.

[23] L. Yang, Q. Lin, X.-Y. Li, T. Liu, and Y. Liu. See through walls with cots rfid system! In *MobiCom*, 2015.

[24] D. Zanetti, B. Danev, and S. Capkun. Physical-layer identification of uhf rfid tags. In *MobiCom*, 2010.