Pyramid: Real-Time LoRa Collision Decoding with Peak Tracking

Zhenqiang Xu, Pengjin Xie and Jiliang Wang
School of Software and BNRist, Tsinghua University, P. R. China
xu-zq17@mails.tsinghua.edu.cn, {xiepengjin, jiliangwang}@tsinghua.edu.cn

IEEE INFOCOM 2021 - IEEE Conference on Computer Communications | 978-1-6654-0325-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/INFOCOM42981.2021.9488695

***Abstract*—LoRa, as a representative Lower Power Wide Area Network (LPWAN) technology, shows great potential in providing low power and long range wireless communication. Real LoRa deployments, however, suffer from severe collisions. Existing collision decoding methods cannot work well for low SNR LoRa signals. Most LoRa collision decoding methods process collisions offline and cannot support real-time collision decoding in practice. To address these problems, we propose Pyramid, a real-time LoRa collision decoding approach. To the best of our knowledge, this is the first real-time multi-packet LoRa collision decoding approach in low SNR. Pyramid exploits the subtle packet offset to separate packets in a collision. The core of Pyramid is to combine signals in multiple windows and transfers variation of chirp length in multiple windows to robust features in the frequency domain that are resistant to noise. We address practical challenges including accurate peak recovery and feature extraction in low SNR signals of collided packets. We theoretically prove that Pyramid incurs a very small SNR loss (< 0.56 dB) to original LoRa transmissions. We implement Pyramid using USRP N210 and evaluate its performance in a 20-nodes network. Evaluation results show that Pyramid achieves real-time collision decoding and improves the throughput by 2.11×.**

***Index Terms*—LPWAN, LoRa, chirp modulation, CSS, collision**

I. Introduction

Nowadays, the need for low power and long range wireless communication arises in many Internet of Things (IoT) scenarios. This results in the birth of a class of Lower Power Wide Area Network (LPWAN) technologies. LPWAN gains more and more popularity due to its energy efficiency and robust communication in the long range. As one of the representative LPWAN technologies, LoRa works in unlicensed bands and promises to connect thousands of distant devices with a single gateway. LoRa promises a high receiving sensitivity and supports long distance communication for up to 10 km. The strong anti-interference ability and high receiving sensitivity of LoRa comes from its CSS (Chirp Spreading Spectrum) modulation. Each LoRa symbol is a cyclically shifted version of a base chirp symbol whose frequency increases linearly with time. Based on the design of the chirp symbol, a receiver can extract a weak LoRa signal with power -20 dB under noise floor by accumulating energy of a whole chirp symbol. Owing to the remarkable performance of LoRa, plenty of IoT applications have already adopted LoRa [1] [2] [3].

However, practical LoRa suffers from collisions in real deployments which severely constrains the network throughput and scalability. Packets from multiple LoRa nodes collide and cannot be decoded. The problem is further aggravated when the network becomes dense and large, which is the supposed application scenario for LoRa. Even worse, LoRa usually adopts lightweight MAC protocols (e.g., Aloha-like MAC in LoRaWAN [4]) to reduce the network complexity and energy consumption. This introduces more packet collisions to LoRa networks. Moreover, using collision avoidance approaches (e.g., CSMA/CA) cannot fundamentally solve this issue considering the complex network environment and a large number of connected nodes.

Existing approaches and fundamental limitations. The application of LoRa technology requires to address the collision problem. Collision decoding and resolution have been studied for a long time and many approaches are proposed. To name a few, mLoRa [5] leverages traditional Successive Interference Cancellation (SIC) for packet decoding. It iteratively decodes the partial clean symbols and then cancels them. mLoRa can support concurrent transmission for up to three nodes. FTrack [6] leverages a simplified time-domain signal SIC. More specifically, FTrack applies Short Time Fourier Transform (STFT) on the time domain signals to separate different packets. The work in [7] proposes a nice design for online concurrent decoding. However, their approach can only decode a very limited number of concurrent packets. Choir [8] leverages the imperfection of LPWAN hardware and improves network throughput. It works mainly on synchronous transmission and the hardware imperfection is difficult to capture under low SNR.

Existing approaches have provided inspiring ideas on decoding collisions in LoRa. A *fundamental limitation* of most existing approaches, e.g., mLoRa and FTrack, is that they focus on time domain signal analysis and do not fully exploit LoRa encoding features. As a result, they require high SNR signal (e.g., $SNR > 0$), while LoRa often has very low SNR (e.g., as low as -20 dB). Meanwhile, most existing approaches incur a relatively high overhead and can only process the received data offline. Thus, they cannot provide online real-time collision decoding.

Our approach. In this work, we propose Pyramid, a real-time LoRa collision decoding approach with low SNR LoRa signals. Pyramid exploits the subtle time offset to separate packets in collisions. Instead of using a single window of signal for decoding, the core of Pyramid is to leverage the variation of chirp length in multiple moving windows. Pyramid separates packets based on a novel method to transfer variation

978-1-6654-0325-2/21/$31.00 ©2021 IEEE

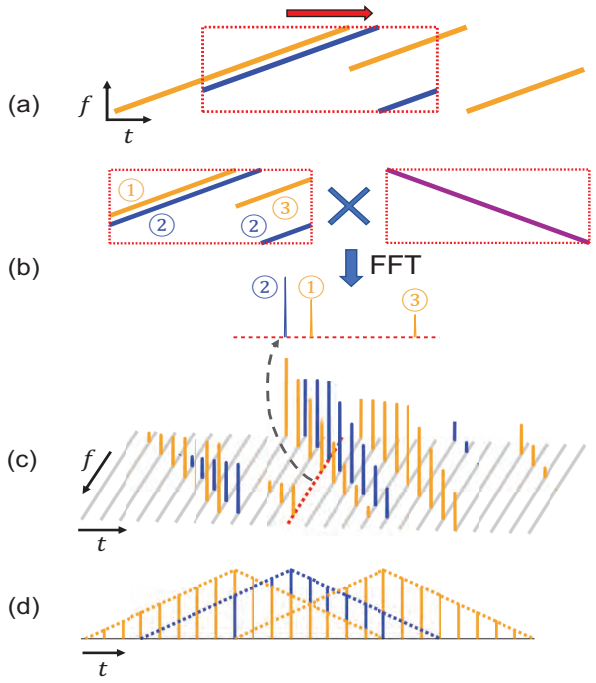


Fig. 1. Pyramid decoding for two collided packets. (a) Two collided packets (the yellow one and blue one) and a decoding window. For illustration simplicity, we hide other surrounding blue symbols. (b) The decoding result for the window in (a) by multiplying a downchirp and applying FFT. There are multiple peaks due to collision. (c) The decoding peaks for a moving window at different time t. (d) The projection of (c) along frequency axis.

of chirp length in multiple windows to robust features in the frequency domain which are resistant to noise. We theoretically prove that our method can lead to a very small SNR loss (< 0.56 dB) to original LoRa transmissions.

To understand how Pyramid works, let us first consider the standard demodulation of LoRa. For simplicity, we can consider that LoRa encodes data using the start frequency of chirps, each with linearly increasing frequency. To decode, LoRa first detects packet preamble and divides the received signal into windows. In each window, a procedure called dechirp is applied that the decoder multiplies the signal with a downchirp symbol with linearly decreasing frequency. This results in a single tone with a frequency indicating the encoded data. By applying Fast Fourier Transform (FFT) over the result, a peak at a specific frequency can be obtained.

Figure 1(a) depicts the collision of two packets, a yellow packet and a blue packet. Assume the decoding window is aligned with one of the packets, as shown in Figure 1(a), there are three chirp segments in each window, one from the aligned packet and the other two from the other packet. This results in three peaks, one blue peak and two yellow peaks, in the frequency domain as shown in Figure 1(b). The key to decode LoRa collisions is to separate chirp symbols for different packets. By continuously moving the window, we can obtain peaks in each window. Figure 1(b) represents the FFT result for the decoding window in different positions. As the window is moving, the time offset between the decoding window and chirp segments accordingly changes, which results in proportional moving of peaks in the frequency domain. Therefore, as shown in Figure 1(c)(d), the peak height for a chirp under a moving window would form a triangle-like trajectory. Further, peaks for different chirps of the same packet form multiple half-intersected triangles. Pyramid exploits those features to group chirp symbols to different packets and decode each packet.

Challenges: The implementation of Pyramid in practice requires to address the following main challenges: (1) The processing is time-consuming and energy-consuming in practice. How to achieve real-time signal processing while obtaining enough features is challenging. (2) Real signal suffers from noise and interference especially under the collided LoRa signal with low SNR. It is non-trivial to accurately measure all peaks in each window. (3) Even when peaks can be measured, the small time offset among packets make the peaks from different packets difficult to separate. (4) Each peak is accompanied by multiple sidelobes. Those sidelobes may distort other peaks and may be misidentified as peaks, leading to decoding errors.

To address the first challenge, we analyze the key processing overhead in the decoding process. We conduct experiments to evaluate the maximum allowable computation on our hardware, and tradeoff the maximum allowable parameters to obtain more features for collided decoding and the process overhead to guarantee real time processing. For the second challenge, we propose an iterative searching and peak cancellation method to accurately find all peaks under low SNR and interference. For the third challenge, we design a multi-level peak group method for packets with small time offset. For the fourth challenge, to remove the impact of sidelobes in the received signal, we propose a data filtering method combining hamming window and rectangle window. Further, we propose a constrained regression method to derive accurate peak information in collisions to reduce the impact of sidelobes.

Main results and contributions:

- We propose Pyramid to decode low SNR LoRa collisions in real-time. The core of Pyramid is to combine signals in multiple windows and transfers variation of chirp length in multiple windows to robust features in the frequency domain which are resistant to noise.
- We address practical challenges for LoRa design in real networks. Further, we theoretically show that our approach can work well with low SNR LoRa signals, and prove that Pyramid incurs a very small SNR loss (< 0.56 dB) to the original LoRa.
- We implement Pyramid in C++ on GNU Radio platform. To the best of our knowledge, this is the first real-time multi-packet collision decoding approach for low SNR signals. We evaluate its performance in a 20-node network. The evaluation results show that Pyramid can achieve real-time LoRa collision decoding and improves the throughput by $2.11\times$ in a real-time environment.

Implementation availability: The source code of Pyramid is available at

https://github.com/jkadbear/gr-lora.

II. Background

A. LoRa Modulation/Demodulation

Modulation. The minimum encoding unit of LoRa is a chirp symbol. Chirp $c(t; f_0, k)$ is a signal with linear changing frequency

$$c(t; f_0, k) = e^{j2\pi(f_0 + \frac{1}{2}kt)t}, 0 \le t < T \tag{1}$$

where f_0 is the start frequency, k is the frequency changing rate, and T is the duration of chirp. The bandwidth of $c(t; f_0, k)$ is $B = |kT|$. LoRa has an important parameter SF to control the chirp shape where $2^{SF} = BT$. SF is a time-bandwidth product and reflects the anti-interference ability of LoRa transmission. Intuitively, a higher SF indicates a large bandwidth B or a long time duration T, implying a higher anti-interference ability. We call $u(t; f_0) = c(t; f_0, k)$ an upchirp when $k > 0$, and call $d(t; f_0) = c(t; f_0, k)$ a downchirp when $k < 0$. Normally, LoRa encodes data with upchirps. LoRa encodes data in starting frequency f_0. There are 2^{SF} start frequencies to use for encoding SF bits data.

More accurately, a LoRa symbol $s(t)$ can be expressed as

$$s(t) = s_1(t)w_1(t) + s_2(t)w_2(t) \tag{2}$$

where $s_1(t) = u(t; f_0)$, $s_2(t) = u(t; f_0 - B)$, $w_1(t)$ and $w_2(t)$ are two rectangular windows. Denote rectangular window $w(t; a, b)$ as

$$w(t; a, b) = \begin{cases} 1, & a \le t \le b \\ 0, & otherwise. \end{cases} \tag{3}$$

Then $w_1(t) = w(t; 0, (\frac{1}{2} - \frac{f_0}{B})T)$ and $w_2 = w(t; (\frac{1}{2} - \frac{f_0}{B})T, T)$.

Demodulation. On the receiver side, the decoder firstly multiplies $s(t)$ with a downchirp $d(t; \frac{B}{2})$. The result is a mixture of two single tones, i.e.,

$$\begin{aligned} s'(t) &= s(t)d(t; \frac{B}{2}) \\ &= e^{j2\pi(\frac{B}{2} + f_0)t}w_1(t) + e^{j2\pi(-\frac{B}{2} + f_0)t}w_2(t). \end{aligned} \tag{4}$$

If we apply Continuous Time Fourier Transform (CTFT) to Eq. (4), those two single tones become two sinc functions that locate at two symmetric positions of frequency $\frac{B}{2} + f_0$ and $-\frac{B}{2} + f_0$. Then we can calculate the peak frequency f_0 to decode the data.

B. LoRa Collisions

In normal LoRa transmission, the decoder extracts the unique peak after dechirping and FFT, and then decode the bits based on the frequency of the peak. When packets collide, there are multiple peaks in a window due to chirps from different packets. The encoded bits from either packet cannot be decoded successfully.

The key advantage for LoRa transmission in low SNR signal is to support low power and long distance transmission. This is mainly benefited from its modulation and demodulation mechanism to aggregate energy in the time domain to a single frequency in the frequency domain. As a result, most existing collision decoding approaches for LoRa, following the traditional principle to decode in the time domain, may not work well for low SNR LoRa signal.

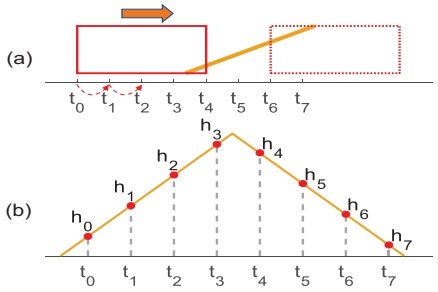


Fig. 2. (a) Sliding window. (b) Peak heights in different decoding windows.

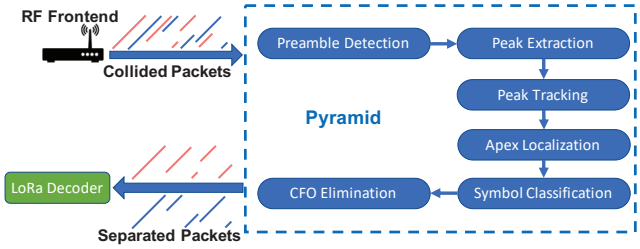


Fig. 3. Pyramid block diagram.

III. Pyramid Design

A. Design Overview

The key for decoding collisions is how to classify multiple peaks into different packets. Our strategy relies on the observations that (1) the peak height of a symbol is proportional to the intersection length of the symbol and the moving window; and (2) while moving the window, the peak location also moves accordingly. As shown in Figure 2, assume the timestamp of the start of a moving decoding window is t. For each timestamp t of the window, we can calculate the peak height for a chirp with respect to window timestamp t. Figure 2 shows the peak height at eight different timestamps (t_0 to t_7). The peak height would reach the highest when the decoding window is perfectly aligned with the symbol. We denote the highest point as *apex* and the corresponding timestamp as *apex timestamp*. Theoretically, the trajectory of peak height forms an isosceles triangle. For apex timestamps of consecutive chirp symbols from the same packet, their timestamp interval should be equal to the chirp length. This is the key feature to group chirp symbols from different packets.

Therefore, the basic idea of Pyramid, as illustrated in Figure 3, works as follows. (1) At first, Pyramid receives signals of collided packets from RF frontend, and detects the existence of LoRa packets based on preamble. (2) Then Pyramid applies a moving window to the received signal

and tries to extract peaks in each window. (3) Pyramid then performs a two-step peak tracking to find the apex. First, Pyramid groups the peaks from the same chirp based on the observation that these peak heights form a triangle trajectory. Then Pyramid calculates the apex timestamp (i.e., position for perfect window alignment) based on a constrained regression method on the peaks of each chirp symbol. (4) Pyramid groups all the symbols in the received signal based on the time difference of apex timestamps. (5) To decode the packet, Pyramid also performs Central Frequency Offset (CFO) elimination and related operations.

We define a window splitting factor q (integer $q > 1$) to control the window sliding step. Suppose the number of samples per chirp symbol is N, the sliding step is $\frac{N}{q}$. Then the trajectory shows as a peak list.

B. Peak Extraction

1) Challenges: Without collision, finding a peak is easy by selecting the FFT bin with the highest magnitude. Collisions introduce two key challenges for peak extraction:

- First, traditionally, peaks can be identified by a predefined threshold. For low SNR LoRa signals, the peak height may be dynamic and easily impacted by interference/noise. This leads to errors in fixed-threshold based peak detection.
- Second, due to limited window size, each peak is accompanied by multiple frequency sidelobes for the FFT result in each window. Those sidelobes may be misidentified as peaks and thus introduce decoding errors.

2) Iterative peak searching: Intuitively, we can find the peaks in each window by a threshold-based method, i.e., finding peaks with a height higher than a predefined threshold. However, this cannot adapt to dynamic signal strength in practice, especially for low SNR LoRa signals. To address the second challenge, we propose a dynamic outlier-based peak search method that finds all peaks by iterative searching and peak cancellation. For each window, we first apply FFT to the dechirping result and find the maximum point as shown in Figure 4(a). Then, we examine whether the maximum point is a real peak by outlier-test. We define a point as an outlier when it is higher than $M + 6\sigma$, where M is the mean of the FFT result, and σ is the variance. If the maximum point is an outlier, then it is considered a peak. Then we cancel the peak from the FFT result with the surrounding points monotonically approaching it as shown in Figure 4(b). Then we iteratively find the peaks according to the above process until no outlier point is found.

3) Sidelobes avoidance: In practice, there exist multiple sidelobes surrounding each peak due to the limited length of the time window. As shown in Figure 4(c), the above peak extraction method may misidentify sidelobes as peaks. To overcome this problem, we apply a hamming window before FFT to suppress the influence of sidelobes. We find that the hamming window is effective in suppressing sidelobes as shown in Figure 4(d). As a side effect, applying hamming window also distorts the height of peaks, which hurts the SNR and changes the shape of the triangle trajectory. To address this problem, we combine the hamming window and rectangle window to derive accurate peaks. We can obtain two peak lists, one from the rectangular window (normal FFT) as shown in Figure 4(c) and the other from the hamming window as shown in Figure 4(d). We select the peaks appearing in both lists and ignore the peaks only appearing in the rectangle window list to avoid sidelobes. The selected two peaks with frequency f_1 and f_2 are shown in Figure 4(c) and Figure 4(d). Meanwhile, we set the peak height as h_1 and h_2 according to the result of the rectangle window for a more reliable peak magnitude.

Summary. Till now, we have found all the peaks in each window, and obtain the information of each peak (t, h, f), where t is the window timestamp, h is peak magnitude, and f is peak frequency. The notion (t, h) that ignoring f also represents a peak when it does not introduce ambiguity.

C. Peak Tracking

Each collided symbol has a series of peaks in consecutive windows. In order to decode the collided messages carried by them, we need to find the peak list belonging to each LoRa symbol.

Linear chirp has a feature that

$$\Delta f = \Delta t \times \frac{B}{T}, \tag{5}$$

where Δt is the time offset and Δf is the frequency offset. The moving interval of the decoding window is set to $\frac{T}{q}$, where q is the window splitting factor. Correspondingly, the peaks will move $\frac{B}{q}$ in the frequency domain. In other words, two peaks in consecutive windows (t_i, h_i, f_i) and $(t_{i+1}, h_{i+1}, f_{i+1})$ for the same symbol should satisfy:

$$\begin{aligned} t_{i+1} - t_i &= \frac{T}{q}, \\ (f_{i+1} - f_i) \bmod B &= \frac{B}{q}. \end{aligned} \tag{6}$$

For each new detected peak, we first check whether it belongs to any known symbol. If the new peak satisfies Eq. (6) with the last peak in a symbol, it is considered as a peak of that symbol. Otherwise, the new peak is from a new symbol. Finally, each symbol can produce a serial of peaks in the moving windows: $peaks = \{(t_0, h_0, f_0), (t_1, h_1, f_1), ...(t_k, h_k, f_k)\}$.

D. Apex Localization

Based on peak tracking, we can find the peaks corresponding to the same symbol. Then we need to find the apex timestamp of the symbol. The interval of apex timestamps for the same packet should be multiples of symbol length T. Therefore, we can classify the symbols into packets with the apex timestamp for each symbol.

Ideally, if we have a continuous moving window, we can locate the apex of the peak list by finding the peak with the highest magnitude. However, a continuous moving window means high computation overhead. In practice, we use a step-based moving window with a window interval of $\frac{T}{q}$. This on one hand can significantly reduce the computation overhead.

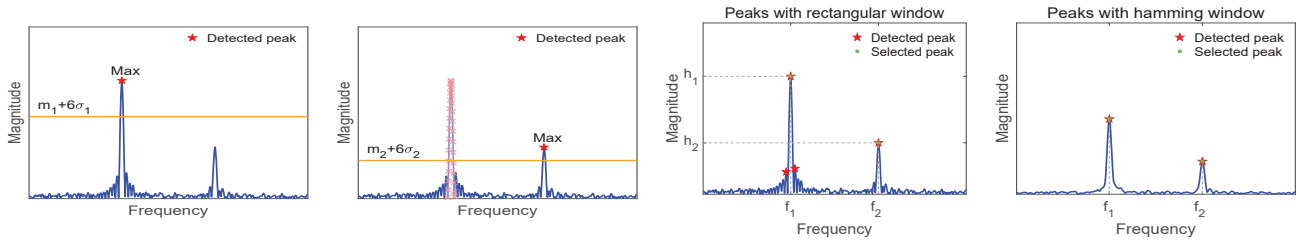


Fig. 4. Peak extraction process. (a) The peak with max magnitude is above $m_1 + 6\sigma_1$. (b) After eliminating the first peak, the second highest peak is above $m_2 + 6\sigma_2$. (c) Peak extraction with rectangular window. (d) Peak extraction with hamming window.

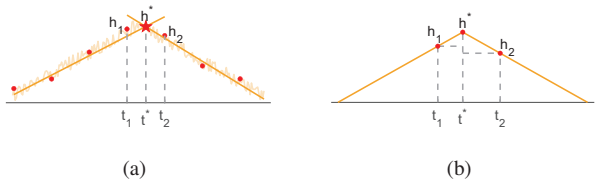


Fig. 5. (a) Linear regression method for calculating the apex under noise. (b) Calculating the apex with top two peaks (t_1, h_1) and (t_2, h_2).

However, on the other hand, this leads to errors in finding the apex. Moreover, the low SNR signal may distort the peak height and leads to larger error in apex finding. To accurately localize the apex, we propose a constrained regression apex finding method based on the peak list. We first divide the peak list into two lists $peaks_1$ and $peaks_2$ based on the top two peaks, i.e., (t_1, h_1) and (t_2, h_2) where $t_1 < t_2$. The list $peaks_1$ consists of (t_1, h_1) and peaks before (t_1, h_1), while $peaks_2$ consists of the remaining peaks. We apply linear regression on $peaks_1$ and $peaks_2$ to find the least square distance. This results in two lines with opposite slopes. Then we can calculate the intersection (t^*, h^*) of the two lines as the apex. Figure 5(a) shows the apex estimation under noise.

As the LoRa signal usually works in very low SNR, it is possible that there is only one peak in $peaks_1$ or $peaks_2$, and the constrained regression apex finding method cannot be used. We design a method to deal with this situation and show that apex can be found even under very low SNR with only two peaks detected. Figure 5(b) shows the neighboring top two points (t_1, h_1) and (t_2, h_2). Assume $p^* = (t^*, h^*)$ is the apex we need to locate. Theoretically, the slope of two triangle edges (i.e., line $\overline{p_1 p^*}$ and $\overline{p^* p_2}$) is fixed and opposite to each other. Leveraging the similar triangle principle, we have

$$\frac{t^* - t_1}{h^* - h_1} = \frac{t_2 - t^*}{h^* - h_2} = \frac{T}{h^*} \tag{7}$$

where T is the duration of a chirp and half of the triangle bottom margin in Figure 5(b). Add the first two parts of Eq. 7 and reserve the third part, we have

$$\frac{t_2 - t_1}{2h^* - h_1 - h_2} = \frac{T}{h^*}. \tag{8}$$

The interval between two peaks is equal to the window interval, i.e.,

$$t_2 - t_1 = \frac{T}{q}. \tag{9}$$

Take Eq. (9) into Eq. (8), we can calculate h^* as:

$$h^* = \frac{h_2 + h_1}{2q - 1} q. \tag{10}$$

From Eq. (8)(10), t^* can be calculated.

When the apex (t^*, h^*) is obtained, we still need a representing frequency bin f^* for symbol decoding. According to Eq. (5), peak frequency

$$f^* = \left(f_1 - t_1 \frac{B}{T} \right) \bmod B. \tag{11}$$

E. Symbol Classification

Based on peak tracking and apex localization, we can find symbols with their corresponding apex (t^*, h^*, f^*). Since the symbol length is T, the interval of apex timestamps for symbols belonging to the same packet should be a multiple of T. Therefore we can classify the symbols into different packets by t^*. Since the maximal distance between two collided packets is $\frac{T}{2}$, we round the time difference between two symbols by $\frac{T}{2}$. When the packet time offset between two packets is small, the symbols from different packets may be classified into the same group. To address this problem, we take energy information of a packet into consideration. First, we check each group to find groups with more than one packet. We sort the symbols in each group. If the apex timestamps form an arithmetic progression and the difference between two consecutive symbols is T, this group is considered as containing only one packet. Otherwise, the symbols in this group need a second step classification. We define the distance between two symbols as

$$D_{i,j} = |h_i^* - h_j^*| \tag{12}$$

After the second step classification, if a group is still considered to contain more than one packet, this group will not be used in the following decoding.

As a result, in this step, we obtain the symbols for each packet sorted by apex timestamp. Each packet has a series of symbols $\{(t_1^*, h_1^*, f_1^*), (t_2^*, h_2^*, f_2^*), ...(t_k^*, h_k^*, f_k^*)\}$. Next we need to decode the message in each packet.

F. Preamble Detection and CFO Elimination

1) Preamble Detection: LoRa preamble is composed of $L_p(6 \sim 65535)$ identical base upchirps. Pyramid detects the existence of a preamble if it finds L_p identical peaks. If Pyramid finds L_p consecutive peaks located at the same position, the preamble of a packet is detected. The procedure for preamble tracking is similar to data symbol tracking in Section III-C. One thing to note is that the trajectory of preamble symbols is a trapezoid instead of a triangle because preamble is a series of base upchirps. The two legs of the preamble trapezoid form a triangle. The preamble apex can be derived from this triangle using the above mentioned apex localization method. Notice that the data apex location has a $0.25T$ difference comparing to preamble apex due to the existence of 2.25 downchirps after preamble in LoRa packets.

2) CFO Elimination: CFO comes from hardware imperfections among different devices. CFO impacts the decoding process, therefore we need to remove CFO before sending the bin values to LoRa decoder. The bin shift caused by CFO is the same for both preamble and data symbols. Since we have already known the composition of preamble, the bin shift caused by CFO can be derived. Suppose we have a record about preamble peak (t_0^*, h_0^*, f_0^*) and one data symbol (t_i^*, h_i^*, f_i^*). After calibrating t_0^* and t_i^*, the amended data bin is

$$f_i^{**} = \left(f_i^* - f_0^* + \frac{B}{4} \right) \bmod B. \tag{13}$$

The constant $\frac{B}{4}$ comes from the existence of 2.25 downchirps.

G. SNR Loss

We analyze the impact of our approach on the original LoRa sensitivity. According to the demodulation mechanism, the sensitivity of LoRa relies on the peak height. In our design, the max peak height in peak list captured by the moving window has at most $\frac{q-1}{q}$ reduction to the apex height. In LoRa, the signal amplitude is proportional to peak height. Translating the reduction to unit dB, Pyramid at most reduces the sensitivity by $s_{loss} = 10 \log_{10} \left(\frac{q-1}{q} \right)^2$. For $q = 8$, $s_{loss} = -1.2$ dB. For $q = 16$, $s_{loss} = -0.56$ dB. In our implementation in §IV-D, we show that $q = 16$ is an available parameter setting for real-time decoding. Therefore, Pyramid has less than 0.56 dB SNR loss comparing to original LoRa.

IV. Evaluation

A. Implementation, Metrics and Setup

We implement an offline mode Pyramid with MATLAB and an online version based on GNU Radio with C++. All the programs run on a laptop (Ubuntu 19.10, i5-7200U, 8GB memory). We first compare the offline Pyramid with other methods. Then we analyze the impacts of different parameters in Pyramid. Finally, we run the online Pyramid to test the real-time performance.

As SNR plays an important role in LoRa decoding, we divide SNR level into 4 categories for evaluation: high SNR ($>$ 15dB), middle SNR (5$\sim$ 15 dB), low SNR (-5 $\sim$ 5 dB) and extremely low SNR ($<$ -5 dB). We mainly adopt three metrics in the evaluation: Symbol Error Rate (**SER**), Packet Reception Rate (**PRR**) and **Throughput**. SER measures the misidentification of LoRa chirp symbol and reflects the raw accuracy of a collision demodulator.

Our experiment devices are shown in Figure 6. The LoRa node is equipped with a solar panel for continuous energy supply. We use USRP N210 as a gateway to receive LoRa signals from LoRa nodes. For the purpose of evaluation, we use an extra beacon node sending signals to control the packet sending process of other nodes and create collisions. Once received a beacon, the nodes wait for a random slot and then send packets. The default setting for the transmission side is (SF12, bandwidth=125kHz, implicit header mode, CodeRate=4/8, CRC on, payload length = 8 bytes) when there is no special statement.

B. Comparing with Existing Works

We compare Pyramid with other state-of-art LoRa collision decoding algorithms, including Choir[8], mLoRa[5], and NScale[9]. Since all three methods only support offline processing, we use the offline mode Pyramid for fair. We create a group of three-nodes collisions to evaluate the performance under different SNR. To finely control the SNR, we manually add noise to the received high SNR signals. Figure 7 shows the SER and throughput of the above four methods.

mLoRa relies on time-domain SIC, therefore its SER increases rapidly when SNR decreases. And it almost cannot work under extremely low SNR. mLoRa tries to capture a non-collided partial symbol, and construct a complete symbol from it. In practice, such a partial symbol can be much shorter than the complete symbol, which results in the imprecision of construction. Therefore the performance of mLoRa seems the worst of these methods. Choir, as reported in other literature, should have a good performance for collision decoding. However, in our results, Choir even has higher SER than mLoRa under high SNR. That is because the assumption adopted in Choir fails in our experiment setting. Choir separates collisions through hardware imperfection and assumes the imperfection is stable. We individually measure the hardware imperfection and find that the fractional bin used in Choir is not quite stable with our nodes. So Choir has bad performance in our evaluation. Considering that LoRa targets at cheap connections, ubiquitous LoRa devices might not provide such stable feature, which could limit the application of Choir. NScale uses non-stationary scaling to decode collisions. Though NScale claims that it has a theoretical SNR loss of at most 1.7dB, the methodology implemented in their paper mainly depends on the partial chirp magnitude ratio. A partial chirp, instead of a complete chirp symbol, cannot fully leverage the high sensitivity of LoRa. Pyramid, however, nearly utilizes the whole chirp symbol energy and can work better under low SNR. For high SNR, linear regression used in Pyramid provides more information than a single scaling ratio used in NScale, which benefits decoding. All these reasons lead to that Pyramid has lower SER and higher throughput.

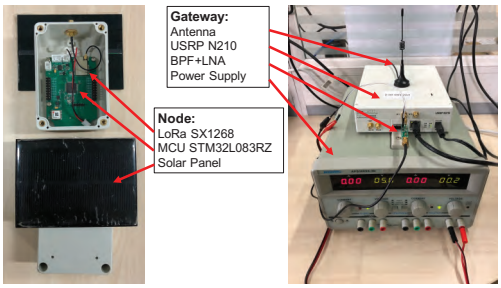
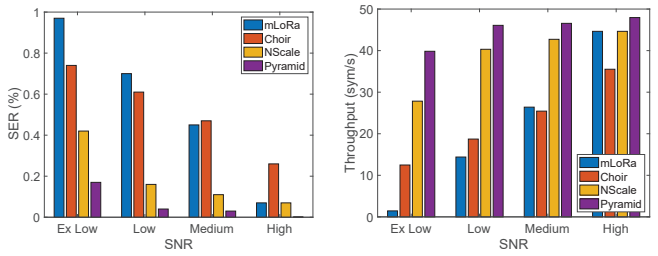
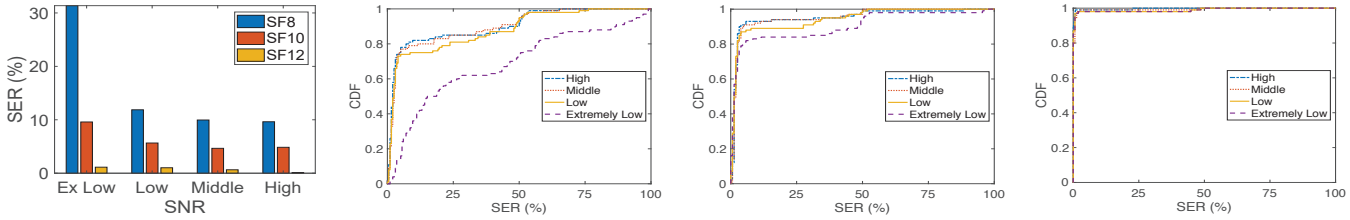


Fig. 6. LoRa nodes and Software-Defined Radio gateway.



(a) SER under different SNR (b) Throughput under different SNR

Fig. 7. Performance comparison between Pyramid and other three LoRa collision decoding methods.



(a) Overall performance (b) SF=8 (c) SF=10 (d) SF=12

Fig. 8. (a) Overall SER performance. (b-d) CDF of SER with different SNRs and SFs.

C. Parameter Impacts

SNR and SF are two main parameters that influence LoRa decoding. We conduct experiments to measure the SER of Pyramid with different SNR and SF. We set SF$= 8, 10, 12$ and generate collisions of two packets under four different SNR levels. Figure 8(a) shows the average SER. Figure 8(b)(c)(d) shows the CDF of SER under different SNRs with SFs respectively. We can see that lower SF leads to higher SER on all SNR levels. This is because lower SF means fewer FFT bins, and the peak frequency of collided symbols from different packets has a higher possibility to coincide with each other due to the arrival time difference and the encoded data. When it happens, the peaks will interfere with each other and lead to errors in apex localization. Though the probability is small that two symbols locating at the same bin, it still can influence the Pyramid's performance when multiple transmissions and packets with long payload exist in the network. We also observe that, although extremely low SNR seems a great challenge to collision decoding, Pyramid can achieve low SER with large SF.

The design of Pyramid indicates that it utilizes the time offset information to classify collided symbols into packets. Thus, small PTO (Packet Time Offset) may incur misclassifications and make it difficult to separate packets. We investigate the influence of PTO on Pyramid performance. Let "$X\%$ PTO" represent that the packet time offset Δt (modulo T) between two packets is $T \cdot X\%$. We denote the PTO of $< 20\%, 20\% \sim 35\%, > 35\%$ as small PTO, medium PTO, and large PTO. The average SERs with different PTO under different SNR levels are shown in Figure 9(a). Figure 9(b)(c)(d) show the CDF of SER with small PTO, medium PTO, and large PTO respectively. From the results, we see that small PTO degrades the performance more under relatively low SNR, but has negligible influence on the performance under higher SNR.

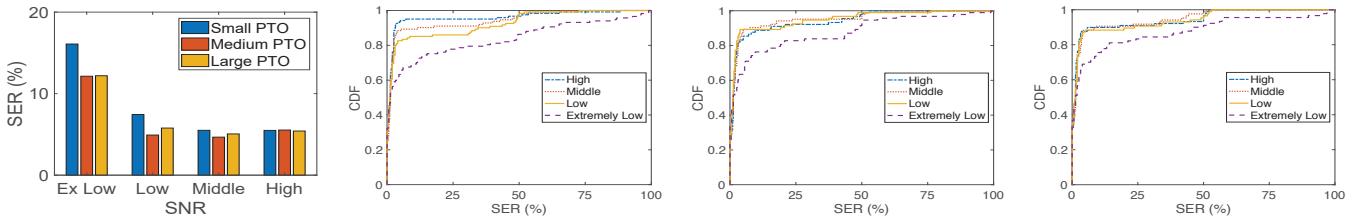
D. Real-Time Collision Decoding

We deploy an indoor 20-nodes network for real-time evaluation.

Maximum Allowable Concurrent Transmissions. First, we test maximum allowable concurrent transmissions that Pyramid supports. Figure 10(a) shows the real-time SER and throughput with collisions of 2$\sim$7 packets. When the number of concurrent transmissions is larger than 6, the throughput starts to decrease because of higher SER. We observe that sometimes Pyramid could correctly extract every symbol in a 6-nodes collision.

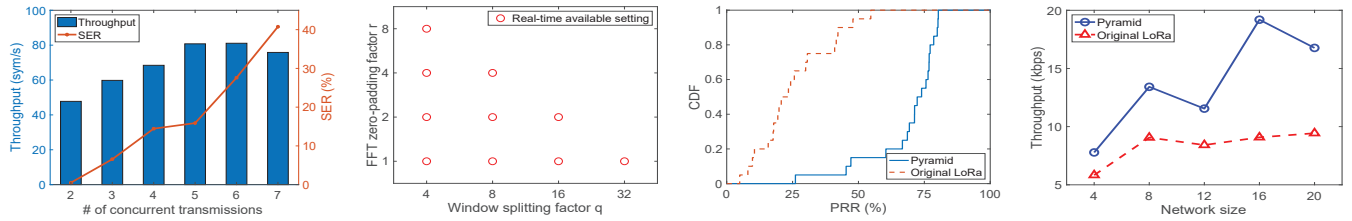
Maximum Allowable Calculation Overhead. In Pyramid design, the most computation-consuming operation is dechirping. The window splitting factor q and the FFT zero-padding factor r are the key parameters determining the major calculation. Though higher q and r provide better resolution, it means larger calculation overhead which could violate real-time requirements. Theoretically, the complexity of q and r is $O(n)$ and $O(n\log(n))$[1] respectively. We conduct experiments to test the maximum allowable calculation overhead for real-time decoding. If the input data rate exceeds the processing ability, GNU Radio will warn us of the unacceptable data loss. The results in Figure 10(c) show all available real-time parameter settings for Pyramid with 1 MHz sampling rate. We see that Pyramid supports at most 32x FFT per symbol without zero-padding or 4x FFT per symbol with 8x

[1]The computational complexity of FFT algorithm is $O(n\log(n))$



(a) Overall performance (b) Small PTO ($< 20\%$) (c) Medium PTO ($20\sim35\%$) (d) Large PTO ($> 35\%$)

Fig. 9. (a) Overall SER performance. (b-d) CDF of SER with different SNRs and PTOs.



(a) Concurrency (b) q and r setting (c) PRR of a 20-nodes size network (d) Throughput of original LoRa and Pyramid in network with different size

Fig. 10. Real-time performance of Pyramid.

zero-padding. q lower than 4 would significantly degrade the collision decoding performance. No zero-padding would result in unstable peak height. In real deployments, $q = 16$ and $r = 2$ seems the best choice and we use them for our experiments.

Real-Time Pyramid v.s. Commodity LoRa. We compare real-time Pyramid with commodity LoRa devices. In this comparison, each node sends a packet with 8 bytes (16 data symbols). One byte indicates the node id and two bytes indicate sequence number. We enable CRC in LoRa physical layer to guarantee the integrity of our packets. If the packet does not pass the CRC check, we mark it as lost. The LoRa decoder outputs the decoding result to a python script, with which we can analyze PRR in real-time. The PRR is calculated conventionally as $num_{recevied}/(seq_{latest} - seq_{first} + 1)$. All the nodes are sending packets at random time with an approximate rate of one packet per 3 seconds. We simultaneously decode the packets with a commodity LoRa node and USRP N210 running Pyramid.

Figure 10(c) shows the CDF of PRR in a 20-nodes network. We see that the PRR of Pyramid is significantly larger than the PRR of the original LoRa, because Pyramid can separate collided packets but commodity LoRa cannot. Figure 10(d) shows the throughput under different network sizes (number of transmitting nodes). For small network sizes like 4 and 8, there are few collisions and Pyramid can easily decode the collisions. Therefore, the throughput increases with the network size. When the network size reaches 12, the successful decoding rate of Pyramid drops a lot because of the dense collisions. Pyramid supports such concurrency and the throughput still grows when network size becomes 16.

The throughput of LoRa receiver stops growing when network size exceeds 8. According to the official document, LoRa could decode the packet with the highest power in collisions if the power difference exceeds 6dB. That is, commodity LoRa receiver has the ability to extract one packet from the collided packets. When the network reaches a certain size (in our experiment, 8), there is a group of nodes owning relatively high power at any time. They hinder the weak signal to be received and stops throughput from growing. Reflected in data, we find that PRR of original LoRa shows large variation. Some nodes even have less than 1% PRR. Pyramid solves this starvation problem by decoding weak signals from collisions. PRR of Pyramid is evenly distributed. Comparing the throughput of Pyramid and original LoRa, Pyramid improves the throughput at most 2.11x with network size 16.

V. Related Work

Low-Power-Wide-Area-Network (LPWAN): Up to now, there are typically two LPWAN categories, i.e. unlicensed LPWANs (LoRa [4], SigFox [10]) and cellular standards targeting LPWANs (NB-IoT, LTE-M [11]). Combining LPWAN with other technologies becomes a hot topic in recent years. PLoRa [12] and NetScatter [13] enable the possibility of long-range backscatter and hundreds of concurrent transmissions from backscatter devices separately by adopting LoRa-like modulation. WIDESEE [3] is able to sense targets in the wide area using LoRa signals. Nandakumar [14] proposes the first localization system consuming microwatts of power by utilizing LoRa chirps. Gadre et al. [15] construct a channel selection model for LoRa and achieves 3.3x data rate improvement over commodity LPWANs.

Despite of low-power, long range and wide coverage of LPWAN, unlicensed LPWAN like LoRa still faces practical challenges towards the ubiquitous connectivity [16]. Packet colli-

sion is one of the major problems requiring to be addressed before bringing LPWAN to large scale deployments [17], [18].

Collision in Wireless Networks: In traditional wireless techniques (e.g. WiFi, Zigbee, etc), existing solutions to collision problems can be divided into two categories, i.e. collision-avoidance and collision-resolution.

In collision-avoidance category, CSMA/CA protocol is widely adopted, which utilizes random backoff time and retransmission to reduce collisions [19] [20]. However, such a method reduces the efficiency of the network and fails to work when there are hidden terminals. RTS-CTS is thus proposed to enhance the performance of CSMA/CA [21]. It overcomes the hidden terminal problem but decreases network efficiency significantly [22].

In collision-resolution category, the capture effect [23] [24] and successive interference cancellation (SIC) [25] are two commonly used schemes. ZigZag [26] proposes a new form of interference cancellation and is able to resolve an m-packet collision. mZig [27] leverages the physical features of Zigbee to decodes multiple packets.

VI. Conclusion

LPWAN technologies have developed rapidly and become an indispensable part to enable fast-growing IoT applications. LoRa, as a representative technology of LPWANs, faces a fundamental problem of performance degradation under packet collisions. Existing approaches cannot provide real-time collision decoding with low SNR LoRa signals. We propose Pyramid, a real-time LoRa collision decoding approach. The core of Pyramid is a novel method to transfer the subtle packet time misalignment, for collided signals of very low SNR, to frequency features resistant to noise. We design the real-time signal process algorithm and show how to address practical challenges such as interference in collisions in Pyramid. We theoretically prove that Pyramid incurs less than 0.56 dB SNR loss comparing to original LoRa transmissions. Pyramid improves the throughput by $2.11\times$ compared with the original LoRa in real-time environment and enables concurrent transmission of 6 LoRa nodes.

Acknowledgements

This work is in part supported by National Key R&D Program of China 2017YFB1003000, National Natural Science Fund for Excellent Young Scholars (No. 61722210), National Natural Science Foundation of China (No. 61932013, 61532012).

References

[1] J-M Martinez-Caro and M-D Cano. IoT System Integrating Unmanned Aerial Vehicles and LoRa Technology: A Performance Evaluation Study. *Wireless Communications and Mobile Computing*, 2019.

[2] Vamsi Talla, Mehrdad Hessar, Bryce Kellogg, Ali Najafi, Joshua R Smith, and Shyamnath Gollakota. LoRa backscatter: Enabling the vision of ubiquitous connectivity. *Proceedings of the ACM IMWUT*, 2017.

[3] Lili Chen, Jie Xiong, Xiaojiang Chen, Sunghoon Ivan Lee, Kai Chen, Dianhe Han, Dingyi Fang, Zhanyong Tang, and Zheng Wang. WideSee: towards wide-area contactless wireless sensing. In *Proceedings of ACM SenSys*, 2019.

[4] LoRaWAN® Specification v1.1. https://lora-alliance.org/resource-hub/lorawanr-specification-v11.

[5] Xiong Wang, Linghe Kong, Liang He, and Guihai Chen. mLoRa: A Multi-Packet Reception Protocol in LoRa networks. In *Proceedings of IEEE ICNP*, 2019.

[6] Xianjin Xia, Yuanqing Zheng, and Tao Gu. FTrack: parallel decoding for LoRa transmissions. In *Proceedings of ACM SenSys*, 2019.

[7] Zhe Wang, Linghe Kong, Kangjie Xu, Liang He, Kaishun Wu, and Guihai Chen. Online Concurrent Transmissions at LoRa Gateway. In *Proceedings of IEEE INFOCOM*, 2020.

[8] Rashad Eletreby, Diana Zhang, Swarun Kumar, and Osman Yağan. Empowering low-power wide area networks in urban settings. In *Proceedings of ACM SIGCOMM*, 2017.

[9] Shuai Tong, Jiliang Wang, and Yunhao Liu. Combating packet collisions using non-stationary signal scaling in LPWANs. In *Proceedings of ACM MobiSys*, 2020.

[10] SigFox White paper. https://www.slideshare.net/fredericengel/sigfox-whitepaper.

[11] Mads Lauridsen, István Z Kovács, Preben Mogensen, Mads Sorensen, and Steffen Holst. Coverage and capacity analysis of LTE-M and NB-IoT in a rural area. In *Proceedings of IEEE VTC*, 2016.

[12] Yao Peng, Longfei Shangguan, Yue Hu, Yujie Qian, Xianshang Lin, Xiaojiang Chen, Dingyi Fang, and Kyle Jamieson. PLoRa: A passive long-range data network from ambient LoRa transmissions. In *Proceedings of ACM SIGCOMM*, 2018.

[13] Mehrdad Hessar, Ali Najafi, and Shyamnath Gollakota. NetScatter: Enabling Large-Scale Backscatter Networks. In *Proceedings of USENIX NSDI*, 2019.

[14] Rajalakshmi Nandakumar, Vikram Iyer, and Shyamnath Gollakota. 3D Localization for Sub-Centimeter Sized Devices. In *Proceedings of ACM SenSys*, 2018.

[15] Akshay Gadre, Revathy Narayanan, Anh Luong, Anthony Rowe, Bob Iannucci, and Swarun Kumar. Frequency Configuration for Low-Power Wide-Area Networks in a Heartbeat. In *Proceedings of USENIX NSDI*, 2020.

[16] Branden Ghena, Joshua Adkins, Longfei Shangguan, Kyle Jamieson, Philip Levis, and Prabal Dutta. Challenge: Unlicensed LPWANs Are Not Yet the Path to Ubiquitous Connectivity. In *Proceedings of ACM MobiCom*, 2019.

[17] Jansen C Liando, Amalinda Gamage, Agustinus W Tengourtius, and Mo Li. Known and unknown facts of LoRa: Experiences from a large-scale measurement study. *ACM Transactions on Sensor Networks (TOSN)*, 2019.

[18] Ferran Adelantado, Xavier Vilajosana, Pere Tuset-Peiro, Borja Martinez, Joan Melia-Segui, and Thomas Watteyne. Understanding the limits of LoRaWAN. *IEEE Communications magazine*, 2017.

[19] Rafael Laufer and Leonard Kleinrock. The capacity of wireless CSMA/CA networks. *IEEE/ACM Transactions on Networking (TON)*, 2016.

[20] Giuseppe Bianchi, Luigi Fratta, and Matteo Oliveri. Performance evaluation and enhancement of the CSMA/CA MAC protocol for 802.11 wireless LANs. In *Proceedings of IEEE PIMRC*, 1996.

[21] Kaixin Xu, Mario Gerla, Sang Bae, et al. How effective is the IEEE 802.11 RTS/CTS handshake in ad hoc networks? In *Proceedings of IEEE GlobeCom*, 2002.

[22] Glenn Judd and Peter Steenkiste. Using emulation to understand and improve wireless networks and applications. In *Proceedings of USENIX NSDI*, 2005.

[23] Kamin Whitehouse, Alec Woo, Fred Jiang, Joseph Polastre, and David Culler. Exploiting the capture effect for collision detection and recovery. In *Proceedings of ACM SenSys*, 2005.

[24] Jeongkeun Lee, Wonho Kim, Sung-Ju Lee, Daehyung Jo, Jiho Ryu, Taekyoung Kwon, and Yanghee Choi. An experimental study on the capture effect in 802.11 a networks. In *Proceedings of ACM WiNTECH*. ACM.

[25] Joseph Blomer and Nihar Jindal. Transmission capacity of wireless ad hoc networks: Successive interference cancellation vs. joint detection. In *Proceedings of IEEE ICC*, 2009.

[26] Shyamnath Gollakota and Dina Katabi. *Zigzag decoding: combating hidden terminals in wireless networks*, volume 38. ACM, 2008.

[27] Linghe Kong and Xue Liu. mzig: Enabling multi-packet reception in zigbee. In *Proceedings of ACM MobiCom*, 2015.