AlignTrack: Push the SNR Limit of LoRa Collision Decoding

Qian Chen, *Graduate Student Member, IEEE*, and Jiliang Wang, *Senior Member, IEEE*

***Abstract*— LoRa has been shown as a promising Low-Power Wide Area Network (LPWAN) technology to connect millions of devices for the Internet of Things by providing long-distance low-power communication when the SNR is very low. Real LoRa networks, however, suffer from severe packet collisions. Existing collision resolution approaches introduce a high SNR loss, i.e., require a much higher SNR than LoRa. To push the limit of LoRa collision decoding, we present AlignTrack, the first LoRa collision decoding approach that can work in the SNR limit of the original LoRa. Our key finding is that a LoRa chirp aligned with a decoding window should lead to the highest peak in the frequency domain and thus has the least SNR loss. By aligning a moving window with different packets, we separate packets by identifying the aligned chirp in each window. We theoretically prove this leads to the minimal SNR loss. In practical implementation, we address two key challenges: (1) accurately detecting the start of each packet, and (2) separating collided packets in each window in the presence of CFO and inter-packet interference. We implement AlignTrack on HackRF One and compare its performance with the state-of-the-arts. The evaluation results show that AlignTrack improves network throughput by 1.68× compared with NScale and 3× compared with CoLoRa.**

***Index Terms*— LPWAN, LoRa, collision resolution, CSS.**

I. Introduction

RECENT years have witnessed the rapid development of Internet of Things (IoTs) technology [1]. LPWANs have been shown as a promising technology to provide low-power long-distance communication for IoT applications [2] such as health monitoring [3], smart agriculture [4], smart traffic light congestion monitoring [5], intelligent parking space allocation [6], etc. LoRa, as a representative LPWAN technology, has attracted both academic and industrial attention in the world [7], [8]. LoRa can communicate for a distance of up to tens of kilometers with very low energy consumption and a very low SNR. The operational lifetime of battery-powered LoRa nodes can reach up to ten years.

However, real LoRa networks suffer from severe packet collisions. The vision of LoRa is to support connections with a large number of low-cost and low-power devices. LoRa network adopts a star network topology for communication. And a LoRa gateway is supposed to connect with thousands of nodes or even more in practice. However, a typical LoRa gateway can only receive LoRa packets from eight channels. This leads to severe packet collisions in real LoRa deployments. Moreover, to reduce control overhead, typical LoRa networks use Aloha [9] based MAC protocols (e.g., LoRaWAN [10]), in which LoRa nodes send packets without detecting the channel status. This feature exacerbates the collision problem in practice [11].

Existing approaches. Different approaches are proposed to address the collision problem for LoRa. mLoRa [12] can decode three collided LoRa packets using Successive Interference Cancellation (SIC). FTrack [13] calculates the continuity of instantaneous frequency to separate collided packets. However, those approaches are based on time-domain signal analysis and do not leverage features of LoRa encoding. Thus, they require a high SNR of the packets (e.g., $SNR > 0$ dB) in decoding, which deviates from LoRa application scenarios. Further, CoLoRa [14] leverages the time offset among packets and translates the time offset to frequency-domain features to separate different collided packets. NScale [15] uses a non-stationary scaling method to translate the time offset to more robust features. Those two approaches can work for the scenario of SNR < 0 dB. However, they need to partition a chirp (symbol in LoRa) and therefore still introduce inevitable SNR loss in practice. In summary, existing approaches require a much higher SNR in decoding collision than traditional LoRa ($SNR \approx -20$ dB). Thus, those approaches cannot work in many practical scenarios.

Our design. To push the limit of LoRa collision decoding under low SNR, we propose AlignTrack, a novel LoRa packet collision decoding approach to minimize SNR loss. Different from the existing approaches that introduce non-negligible SNR loss and require a higher SNR than traditional LoRa. AlignTrack leverages the entire chirp to concentrating the whole energy and can decode collisions in the SNR limit of the original LoRa ($SNR \approx -20$ dB).

LoRa modulates data with chirps of linearly *increasing* frequency with different frequency shifts to encode data bits. The start frequency shift f_s is decoded by de-chirp: a linearly *increasing* chirp is multiplied with a standard linearly *decreasing* down-chirp, which leads to a single tone at frequency f_s. By applying Fast Fourier Transform (FFT) to the de-chirp result, we can estimate the start frequency shift f_s.

AlignTrack is based on the following basic findings. Figure 1 shows three decoded packets with time offset τ_1

Manuscript received 5 April 2022; revised 4 November 2022; accepted 22 December 2022; approved by IEEE/ACM Transactions on Networking Editor K. C.-J. Lin. Date of publication 23 January 2023; date of current version 17 October 2023. This work was supported in part by NSFC under Grant U22A2031, Grant 61932013, and Grant 62172250. *(Corresponding author: Jiliang Wang.)*

The authors are with the School of Software, Tsinghua University, Beijing 100084, China (e-mail: cq20@mails.tsinghua.edu.cn; jiliangwang@tsinghua.edu.cn).

Digital Object Identifier 10.1109/TNET.2023.3235041

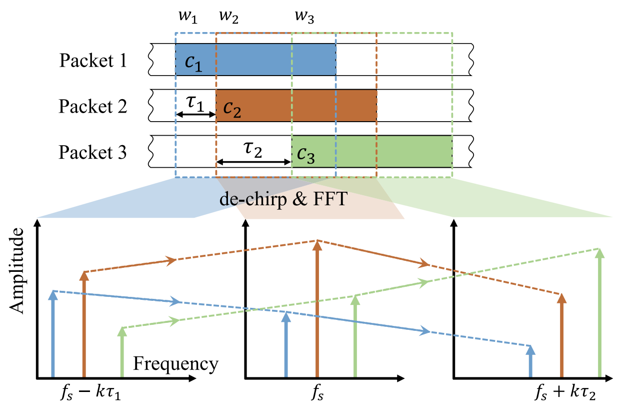

Fig. 1. An example of AlignTrack to decode a 3-packet collision. AlignTrack moves a window to align with different chirps, and finds the aligned chirps to separate packets.

and τ_2. We only show a chirp for each packet for brevity. Assume we have a moving window aligned with the chirps of those three packets. The length of the moving window is equal to the length of a chirp. Figure 1 shows the decoding result of three windows w_1, w_2, and w_3 aligned with three chirps c_1, c_2, and c_3 correspondingly. We can see multiple peaks in each window due to multiple chirp segments, leading to decoding failure.

The key step in collision decoding is to separate those peaks in each moving window to different packets. The main idea of AlignTrack is to pick out the peak of the chirp *aligned* with the moving window and thus separate peaks for different packets based on those aligned chirps.

We first detect the start of each packet and align a moving window with each packet. A chirp in each packet will appear in three consecutive windows. And thus there will be three peaks in those consecutive windows corresponding to the same chirp. We find that the frequency of those three peaks is proportional to the time offset between the chirp and the window. For example, chirp c_2 leads to three peaks in window w_1, w_2, and w_3 with frequency $f_s - k\tau_1$, f_s, and $f_2 + k\tau_2$, respectively. Therefore, we can group peaks corresponding to the same chirp in consecutive windows by calculating the frequency shift. Meanwhile, we find that the height of those three peaks is proportional to the length of chirp segment in the window. The peak of a chirp in different windows reaches its highest when the chirp is aligned with the window. Therefore, we can pick out which group of peaks is corresponding to the chirp aligned with the window by comparing the height of the peaks in a group. AlignTrack leverages the frequency and height characteristics to determine the peak of the chirp aligned with each window and then separate packets. For example, AlignTrack first groups peaks at $f_s - k\tau_1$, f_s, and $f_s + k\tau_2$ in window w_1, w_2, and w_3 for the same chirp, and then determine the peak at f_s corresponds to the chirp c_2 aligned with window w_2 *iff* (if and only if) the heigh of the peak at f_s is the highest compared with the other two peaks. Then, we can separate peaks to different packets based on the aligned chirps and decode those packets.

Challenges. The practical implementation of AlignTrack faces three non-trivial challenges.

(1) How to find the accurate start of each packet in the collided signal under the impact of Central Frequency Offset (CFO)? Intuitively, we can detect the start of a packet based on the preamble. However, this leads to non-negligible errors due to the impact of CFO and inter-packet interference. Moreover, traditional CFO recovery may fail to work due to collisions. We leverage the preamble and SFD in each packet. The preamble contains baseline up-chirps of linearly increasing frequency, while SFD contains baseline down-chirps of linearly decreasing frequency. CFO introduces the same frequency shifts to preamble and SFD, and we can estimate the CFO by combining the up-chirps and down-chirps. Further, we find the above CFO estimation method fails in collided packets due to the challenge of finding preamble and SFD that are for the same packet. We propose a method to identify up-chirps and down-chirps that belong to the same packet, and thus estimate the accurate CFO in collisions.

(2) How to accurately detect all peaks under low SNR? The height of peak for a low SNR signal may be close to the noise. Thus, using a pre-determined height threshold may fail to identify all peaks. Moreover, a peak may be surrounded by points with similar height due to the limited FFT resolution, and these points may be mis-identified as peaks. We design an iterative peak search method to find all peaks in each window. In each iteration, we calculate a dynamical threshold based on the statistic information of existing FFT results in the window. Then we find the highest point and compare its height with the threshold to judge whether it is a peak. For each peak, we remove the points before/after it from the FFT result to avoid identifying those points as peaks. The iteration terminates when no peaks are found.

(3) How to alleviate the interference among peaks and recover the precise peak information (position and height)? For example, the sidelobes of one peak can distort the position and height of other peaks. Moreover, due to the near-far problem, sidelobes of strong signals can be higher than peak of weak signals, leading to mis-identified peaks. Typically, a filter (e.g., Hamming window) can be applied to the received signal to reduce the amplitude of sidelobes. However, there are still high sidelobes after filtering. Further, we find that sidelobes are symmetric with the real peak and exploit this to iteratively remove those symmetric sidelobes.

Main results and contributions. The main results and contributions of this work are as follows.

- We propose AlignTrack to push the limit of decoding low SNR LoRa packet collision. AlignTrack leverages the entire chirp in LoRa, and thus introduces very small SNR loss while existing approaches introduce non-negligible SNR loss due to the use of partial chirp. In principle, AlignTrack will not degrade the performance of traditional LoRa.
- We address non-trivial practical challenges such as inter-packet interference and the impact of CFO in the practical design of AlignTrack. We make full use of the structure of LoRa preamble and SFD.
- We implement AlignTrack on the HackRf One platform. AlignTrack sits at the gateway side and can decode collisions for COTS LoRa end nodes without any hw/sw

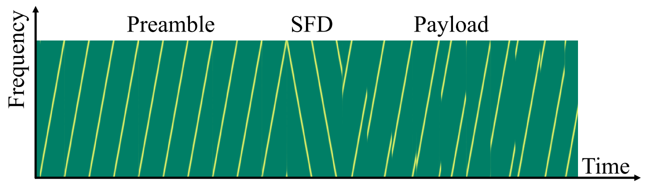

Fig. 2. LoRa packet structure.

change. The evaluation results show that AlignTrack improves the network throughput by 1.68× compared with NScale [15] and 3× compared with CoLoRa [14].

The rest of the paper is organized as follows. Section II introduces the background of LoRa. Section III discusses key observations of our design and limitations of existing approaches. Section IV shows the design of AlignTrack. Section V discusses the solutions in different collision situations. Section VI presents the implementation and evaluation results. Section VII discusses the related work and Section VIII concludes our work.

II. Background

A. Chirp Modulation/Demodulation

LoRa modulates signals using the Chirp Spread Spectrum (CSS). The basic symbol is a chirp with linearly increasing/decreasing frequency occupying a certain bandwidth. Therefore, the anti-interference ability and the resistance to the multipath and Doppler effects can be strong. A baseline chirp is represented as $C(t) = e^{j2\pi(f_0+\frac{1}{2}kt)t}$, where $|k| = BW/T_{chirp}$ is the frequency changing rate, $f_0 = -BW/2$ is the start frequency at time 0, and T_{chirp} is the time duration. T_{chirp} is usually determined by the Spreading Factor (SF) and frequency bandwidth (BW), i.e., $T_{chirp} = 2^{SF}/BW$. Note that in real LoRa network, T_{chirp} is determined by the SF and the frequency swing of the signal, which is usually slightly smaller than BW. A chirp is an up-chirp when $k > 0$, and otherwise is a down-chirp ($k < 0$). The frequency of a baseline up-chirp linearly increases from $-BW/2$ to $BW/2$.

CSS modulates data bits by shifting the start frequency f_0 to $f_0 + f_s$, i.e.,

$$C(t, f_s) = C(t)e^{j2\pi f_s t} \tag{1}$$

where f_s is the frequency shift to encode data bits. Note that the frequency above $BW/2$ is folded to $-BW/2$ to fit in the range $[-BW/2, BW/2]$.

To demodulate $C(t, f_s)$, LoRa first multiplies it with the base line down-chirp $C^{-1}(t)$ ($C^{-1}(t)$ is the conjugate of the $C(t)$). This de-chirp operation can concentrate the signal power in the time domain to a single frequency, and we obtain

$$C(t, f_s)C^{-1}(t) = e^{j2\pi f_s t} \tag{2}$$

By applying FFT to the result, we can derive f_s and decode the chirp.

B. LoRa Packet Structure

As shown in Figure 2, a LoRa packet is usually comprised of three parts: preamble, start frequency delimiter (SFD), and payload. The preamble contains a variable preamble of 6∼65535 baseline up-chirp to determine the start of the LoRa packet and a sync word of 2 up-chirps to differentiate the LoRa network [16]. The SFD contains 2.25 baseline down-chirps to indicate the start of the payload. The payload contains data bits modulated with up-chirps.

C. LoRa Packet Collisions

When there is no collision, the de-chirp and FFT operation in each window results in a single peak which encodes the data bits. LoRa can demodulate chirp by piking the unique peak and decode data according to the frequency of the peak. When collision happens, LoRa packets are mixed together in the time domain, and the de-chirp operation results in multiple peaks. It is difficult to separate those peaks and decode the collided packets. And thus there comes the problem that how can we judge the corresponding relationship among peak, chirp signal and LoRa packet.

III. Motivation

In this section, we show the basic knowledge and main idea of how to extract right peak from collision.

A. Traditional LoRa Decoding

The complete workflow of LoRa receiver is as follows. When a LoRa receiver is awake in scheduled TX/RX windows and operates in the receive mode, the LoRa receiver will continuously samples the given channel to detect, receive and decode the incoming signal. As for signal without collision, the LoRa receiver aligns the chirp symbol with a moving window (the smallest decoding unit containing the LoRa signal) that has the same length T_{chirp} as a chirp signal. Every time the moving window moves a fixed step T_{chirp} to match all chirps. The LoRa receiver then multiplies a down-chirp with the aligned chirp symbol, adopts FFT on the result to derive a unique peak, and transforms the frequency of the peak to data. For a single chirp symbol appeared in the moving window, the LoRa receiver can simply choose the highest point as the unique peak.

B. Basic Observations in Collision

For multiple collided packets, we first detect the start of each packet. Then, we apply a moving window aligned with each packet to the signal. We align the moving window with all chirps in the received signal as traditional LoRa while the moving step is determined by the time offset among collided packets. As shown in Figure 3, we consider three consecutive windows, each containing multiple partial chirp segments and a particular aligned chirp (i.e., a chirp completely included in the window). The aligned chirp in the middle window leads to the peak with local maximum amplitude after de-chirp and FFT. Meanwhile, a portion of this chirp is also contained in its preceding window and the following window, leading to two corresponding peaks. We first show the frequency constraint of peaks for the same chirp in consecutive windows.

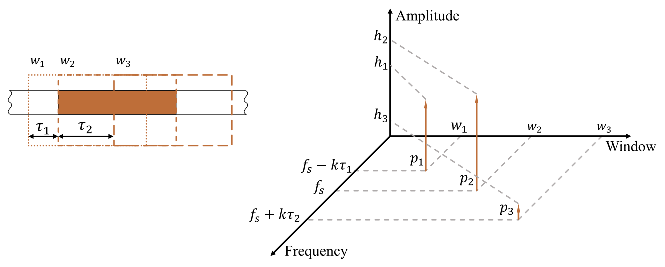

Fig. 3. The peak frequency and height in the moving window.

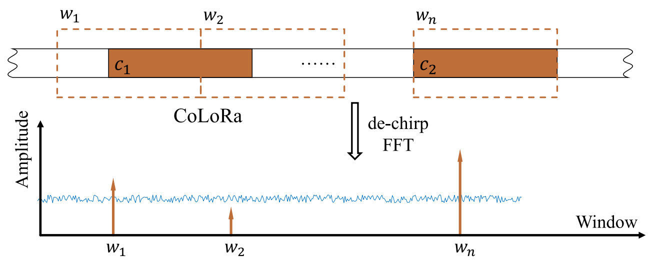

Fig. 4. The SNR loss comparison for different methods. Our method uses the entire chirp while existing methods use partial chirps.

Assume the chirp aligned with the middle window w_2 is $R(t) = A \cdot C(t, f_s)$, where A is the signal amplitude and f_s is the start frequency shift. The segment of $R(t)$ contained in window w_1 can be written as

$$\begin{aligned} r_1(t) &= R(t - \tau_1) \\ &= Ae^{j2\pi f_s(t-\tau_1)} C(t - \tau_1), t \in [\tau_1, T_{chirp}) \end{aligned} \tag{3}$$

The time offset can be translated to the frequency shift, i.e., $C(t - \tau_1) = e^{-j2\pi k\tau_1 t} C(t)$. Thus, we have

$$\begin{aligned} r_1(t) &= Ae^{j2\pi(f_s(t-\tau_1)-k\tau_1 t)} C(t) \\ &= Ae^{-j2\pi f_s \tau_1} e^{j2\pi(f_s - k\tau_1)t} C(t), t \in [\tau_1, T_{chirp}) \end{aligned} \tag{4}$$

Thus, the frequency of the peak p_1 for $r_1(t)$ in w_1 is

$$f_1 = f_s - k\tau_1 \tag{5}$$

Then we move the window distance τ_1 to align with $R(t)$ and the frequency of p_2 should be $f_2 = f_s$. Continue moving the window distance τ_2 to align with the next chirp in received collided signal. The segment of $R(t)$ contained in window w_3 can be written as

$$\begin{aligned} r_2(t) &= R(t + \tau_2) \\ &= h * e^{j2\pi f_s \tau_2} * e^{j2\pi(f_s + k\tau_2)t} C(t), t \in [0, \tau_2) \end{aligned} \tag{6}$$

And peak p_3 shows at the frequency of $f_3 = f_s + k\tau_2$. *Frequency Constraint*: The frequency shift of three peaks p_1, p_2, and p_3 corresponding to the same chirp, is proportional to the time shift between the moving window and the chirp. This indicates that based on the frequency constraint, we can find the peaks corresponding to the same chirp in consecutive windows.

Denote h_1, h_2, and h_3 as the height of peaks in window w_1, w_2, and w_3. Thus h_1 can be calculated as

$$h_1 = \sum_{n=0}^{N-1} r_1[n] e^{-j2\pi \frac{nT_{chirp}}{N}} = A(T_{chirp} - \tau_1) S_r \tag{7}$$

where S_r is the sampling rate and N is the total sample points of $r_1(t)$, i.e., $N = (T_{chirp} - \tau_1) S_r$. Similar, h_2 and h_3 can be calculated as

$$\begin{aligned} h_2 &= A T_{chirp} S_r \\ h_3 &= A(T_{chirp} - \tau_2) S_r \end{aligned} \tag{8}$$

Height Constraint: The height of the peak is proportional to the chirp segment length, i.e.,

$$h_1 : h_2 : h_3 = T_{chirp} - \tau_1 : T_{chirp} : T_{chirp} - \tau_2. \tag{9}$$

The peak height reaches the highest when the chirp is aligned with the window. This indicates that based on the height constraint, we can find the peak of the aligned chirp by comparing the height of peaks corresponding to the same chirp.

C. Key Idea

In tradition, the moving step of a moving window is fixed. Traditional LoRa receiver decodes LoRa packets by shifting a moving window with moving step T_{chirp}. Similarly, existing methods such as Choir, CoLoRa and NScale shift a moving window with fixed step T_{chirp}. These methods simply use a non-overlapped moving window and do not consider the relationship among time offset, window offset and frequency offset.

AlignTrack uses the moving window differently. The moving window does not move with a fixed step. AlignTrack adopts a moving window to align with all chirps in the received signal, which means the moving step is dynamic and determined by the time offset among different collided packets. When there is an m-packet collision, there will be different moving steps in m.

D. Limitations of State-of-the-Art Methods

Existing methods such as FTrack [13] and mLoRa [12] use time-domain signal amplitude to identify collided packets. Their methods can mainly work for high SNR (e.g., $SNR > 0$ dB). This is also shown in their experiments [12], [13].

Further, CoLoRa [14] and NScale [15] propose to leverage features in frequency domain to identify collided packets. However, they use a portion of a chirp instead of the entire one for collision resolution, which introduces non-negligible SNR loss. For example, as shown in Figure 4, CoLoRa partitions a chirp into two parts and leverages the ratio of height between those two parts to decode collisions. It proposes a method to guarantee that both parts are larger than 1/3 of the entire chirp. To decode a chirp, the peak after de-chirp should be higher than noise. The height of a peak is mainly determined by the length of the chirp segment in a window. The longer the length of the chirp segment, the more energy will be concentrated, and the higher the peak will be. When the peak of an entire chirp is higher than the noise, the peak height in CoLoRa, which corresponds to only a portion of the chirp, can be under the noise. Thus, the peak cannot be identified, and the collision

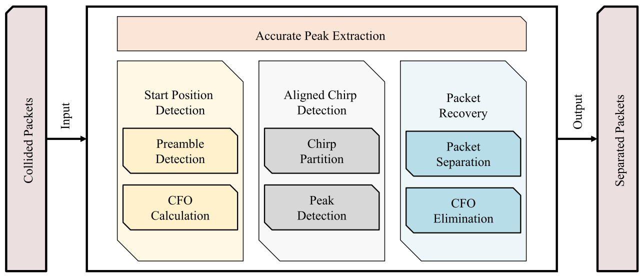

Fig. 5. Overall design of AlignTrack.

cannot be decoded. In such a case, CoLoRa cannot work even when the packet has a high SNR. AlignTrack leverages the entire chirp in aligned window for LoRa to concentrate all the energy of a chirp to retain the peak height to the greatest extent to resist noise, and thus only introduces very small SNR loss with existing approaches.

E. Summary

Based on the analysis above, we can see the relationship between chirp peak and position offset between window and chirp.

- Collisions result in multiple peaks in a window. The key step in collision decoding is to separate those peaks into different packets. The state-of-the-art collision decoding approaches in LoRa have a high SNR loss as they use parts of a chirp to separate peaks.
- For consecutive windows on the collided signal, the peaks for the same chirp in different windows have peak frequency offset proportional to window offset.
- The peak height reaches the highest when the chirp is aligned with the window. For each peak in the window, we can determine whether it corresponds to the aligned chirp as follows. First, we find the peaks corresponding to the same chirp in consecutive windows. Then, the peak corresponds to the aligned chirp *iff* the peak is highest compared with the peaks in the preceding window and following window.

In short, when collision happens, there are more than one chirp in a sliding window. After de-chirp and FFT, we can get more than one peak. Compared with the corresponding peak of the same chirp in adjacent window, the peak should be the highest when chirp is aligned with sliding window.

IV. AlignTrack Design

A. Overview

As shown in Figure 5, the design of AlignTrack mainly consists of the following steps:

1) Start Position Detection: For the received collided signal, we first detect the number of collided packets by adopting a preamble detection method [17], [18], and then calculate the accurate start position of each packet by calculating and eliminating the influence of CFO.
2) Aligned Chirp Detection: After knowing the start position of each packet, we calculate the positions of all chirps in received signal and apply a moving window to align with these chirps in chirp partition step. Then, by comparing the peak information (frequency and height) with adjacent windows, we find the peak of the chirp aligned with the window in peak detection step.
3) Packet Recovery: We first group peaks of aligned chirps and separate them into different packets based on the position of each chirp. Then, we eliminate the impact of CFO to decode peaks of each packet.

Among these steps, the most basic and important part is to extract all peaks with frequency and height information in each window. And thus we need to adopt an accurate peak extraction method to help calculating the start position, demodulating the aligned chirp, and recovering packets.

B. Challenges

- How to recover all peaks in each window as the chirps for different collided packets will interfere with each other?
- How to find the accurate start for all collided packets?
- How to find the peak of the aligned chirp in each window?

C. Peak Extraction

When there is no collision, there is only one chirp in a demodulation window, and we can obtain a single and accurate peak by simply get the highest point after de-chirp and FFT. However, when collision happens, there are multiple chirps (suppose there are N chirps in a window) with different energy level in a window, and there are N peaks with different height after de-chirp and FFT. Due to the energy difference between different packets, the height of a peak from low energy packet can be lower than the height of a sidelobe from high energy packet. Therefore, we cannot directly select the highest N points as peaks. Collision brings the following challenges in peak extraction.

1) Typically, peaks can be identified by a pre-determined threshold. Due to the uncertainty of packet time offset and the variation of signal strength and SNR, the height of peaks for different packets can vary significantly. A fixed pre-determined threshold cannot work.
2) Due to the limited resolution of FFT, the accurate frequency of the peak is difficult to derive, and the height for surrounding points of a peak is also high.
3) The sidelobes of a peak can distort the position and height of other peaks. Even worse, the sidelobes of high peaks may be mis-identified as peaks.

1) Iterative Peak Extraction: To address challenges 1 and 2, we propose an iterative peak extraction method by adopting a dynamic threshold. Actually, peak must be much higher than most data points in FFT window, i.e. the height of peak deviates from the normal height of data. we choose a dynamic threshold according to the height of all points after de-chirp and FFT to address the challenge 1. We first calculate a higher threshold to extract peaks with higher energy. After extracting this peak and removing the high points around the peak (challenge 2), we then calculate a lower threshold to extract peaks with lower energy.

Denote the result of FFT as $H[i]$ $(1 \leq i \leq N)$ where H is the amplitude, and N is the total number of points in FFT. In each iteration, we first find the highest peak $H[i_m]$ in $H[\cdot]$. Then, we need to determine whether it is a real peak based on the combination of $mean(H)$ and $std(H)$, where $std(H)$ is the standard deviation of H. In traditional outlier detection algorithm, data points that exceed $mean(H) + k \cdot std(H)$ (k = 3) are considered as outliers. We find that a larger k leads to more false negative peaks, and a smaller k leads to more false positive peaks. We evaluate the performance for different k and find that $k = 6$ leads to the best performance in peak identification. Thus, we choose $r = mean(H) + 6 \cdot std(H)$ as the threshold. If $H[i_m] < r$, the iteration terminates. If $H[i_m] \geq r$, we set the point at i_m as a peak and add i_m to the peak array I. Due to limited frequency resolution in FFT results, the points surrounding i_m may also have a high height and can be mis-identified as peaks in following iterations. We remove those surrounding points as follows. We find the closest local minimum before and after i_m, i.e., $H[i_m - a]$ and $H[i_m + b]$. Then, we remove all points between $i_m - a$ and $i_m + b$ from H. In the next iteration, we update r based on the remaining points in H. It should be noted the threshold $r = mean(H) + 6 \cdot std(H)$ is dynamic according to H. The threshold r varies in each iteration as H varies. After iteration, we can get a set of outliers from FFT window and then we choose peaks from them by eliminating the influence of sidelobes.

2) Sidelobe Elimination: Section IV-C.2: The sidelobes are mainly caused by two factors. (1) The spectral leakage which is related to the FFT operation. (2) The out of synchronization of the received signal. The FFT operation causes the spectral leakage [19], and thus there are sidelobes to be eliminated after de-chirp and FFT. To improve the resolution and increase the accuracy of demodulation, we apply the zero-padding method on FFT, which further increase the influence of sidelobes. Moreover, when the demodulation window aligns with one chirp in the received signal, there are also many unaligned chirps in the window due to the collision. The out of synchronization of these unaligned chirps will also cause the sidelobes [20], [21]. To address challenge 3 (the influence of sidelobes), a straightforward method is to apply a Hamming filter to the received signal to reduce the amplitude for the sidelobes of each peak. However, the remaining high sidelobes (e.g., sidelobes of very high peaks) can still be mis-identified as peaks. We find that sidelobes are symmetric around a certain peak in terms of frequency and height. Based on this, we design the following method to remove sidelobes. In practice, due to the impact of noise and limited FFT bin resolution, the frequency (height) of two symmetric sidelobes cannot be exactly the same. Therefore, we derive symmetric sidelobes as follows: If two peaks have similar height and frequency, we consider them as symmetric peaks. We sort the peak array I in ascending order of height. For each peak i in I, we find if there exist symmetric peaks centered at i. If yes, we remove those symmetric peaks from I. Finally, we use the remaining peaks in I as the extracted peaks. The detailed algorithm of peak extraction is shown in Algorithm 1.

Algorithm 1 Peak Extraction

Input: H: the amplitude result of FFT
Output: I: the index array for peaks after removing sidelobes

```
1: while true do
2:     $i_m = arg_i max(H)$;
3:     $r = mean(H) + 6 \cdot std(H)$; //in each iteration, recalculate $i_m$ and $r$ according to a new $H$
4:     if $H[i_m] > r$ then
5:         add $H[i_m]$ to $I$
6:         find closest local minimum before and after $i_m$, i.e.,$i_m - a$ and $i_m + b$;
7:         remove points between $i_m - a$ and $i_m + b$ from $H$; //$H$ is changed after each iteration
8:     else
9:         break;
10:    end if
11: end while
12: sort I by ascending order of their peak height;
13: while $i < I.length$ do
14:     for $j = i + 1$; $j$++; $j < I.length$ do
15:         find $k$ such that $I[j] - I[i] == I[i] - I[k]$; //frequency symmetric at $i$
16:         if $H[I[k]] == H[I[j]]$ //height symmetric then
17:             set isSidelobe[k] and isSidelobe[j] to TRUE;
18:         end if
19:     end for
20:     $i = i + 1$;
21: end while
22: I = I[isSidelobe$\neq$ TRUE];
```

According to above accurate peak extraction method, we can get all the peaks information in FFT window accurately. Combined with position of the moving window, we can use the information to detect preamble and decode chirps.

D. Start Position Detection

The key idea of AlignTrack is to align with all chirps in the received signal, and thus we need to know the accurate position of chirp. We leverage the structure (preamble and SFD) of the LoRa packet to detect the number of collided packet and the accurate start position of each packet in a collision. Then, we can get the position of all chirps as the length of each chirp can be calculated by the already known SF, BW and sampling rate.

1) Preamble Detection: The preamble of a LoRa packet consists of N_p (6∼65535) baseline up-chirps with $f_s = 0$. As shown in Figure 6, we apply a non-overlapped moving window to the received signal with moving step T_{chirp}. In each window, we multiply it with a baseline down-chirp and calculate the FFT result. For the preamble, we should have N_p peaks of the same frequency in N_p consecutive windows.

In Figure 6, c_1, c_2, and c_3 are three baseline up-chirps in a preamble. Assume the time offset between the start of the moving window and the start of the packet is τ. Given two windows w_1 and w_2, the peak for the segment of a chirp c_1 in w_1 is $f_1 = f_s - k\tau$, and the peak for the segment of chirp

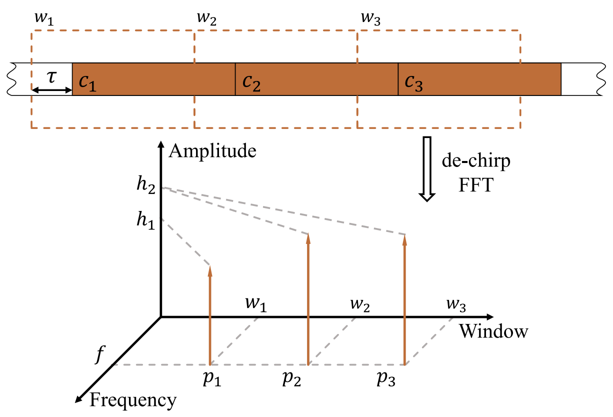

Fig. 6. Preamble detection process: the peaks of baseline up-chirps in preamble appear at the same frequency.

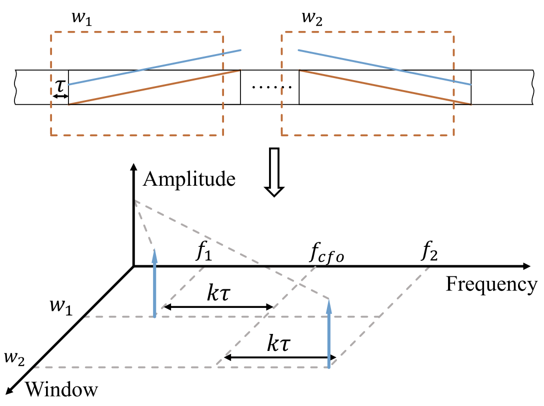

Fig. 7. Calculate CFO by combining preamble and SFD.

c_1 in w_2 is $f_2 = f_s + k(T_{chirp} - \tau)$. $kT_{chirp} = BW$ and the frequency of peak varies from 0 to BW. Thus, f_2 should be $f_2 = f_s - k\tau$. Similarly, the frequency of peak for the segment of chirp c_2 in w_2 should be equal to the frequency of peak for the segment of chirp c_1 in w_1 as they share the same start frequency shift and time offset between moving window and the chirp.

We can find that 1) the peaks of segments corresponding to the same chirp in two adjacent windows has the same frequency; 2) the peaks of two segments of two baseline up-chirps has the same frequency in the same window, and thus there will be only one peak after de-chirp and FFT. Thus, there will be only one peak with frequency $f = f_s - k\tau$ in the window containing baseline up-chirps in a LoRa preamble. We can find the preamble by finding N_p consecutive peaks at the same frequency f. When there are multiple collided packets, we can find multiple groups of peaks, each group corresponding to a packet. And thus we can get the number of collided packets in the received signal. Note that the existing works [22], [23] mention that a preamble can be detected if at least four chirps of the preamble do not get destroyed. However, there may be consecutive identical chirps of unknown length in the payload of the LoRa packet. And if we find the preamble by finding four consecutive peaks at the same frequency, we may mistake these chirps as the preamble and make errors in LoRa packet demodulation. By finding N_p consecutive peak, we can minimize the occurrence of such errors. When there is no CFO, we can simply calculate the start position of each packet after knowing the position of moving window and the frequency f_s.

2) CFO Calculation: In practice, the existence of CFO introduces errors in packet start detection, which will influence the calculation of the accurate start position. And thus, we need to estimate the influence of CFO under collision and eliminate it.

Without Collision We first consider the case without collision. We use both the preamble (baseline up-chirps) and SFD (baseline down-chirps) in each LoRa packet to estimate CFO [14]. Figure 7 shows a pair of baseline up-chirp and down-chirp without CFO (brown line) and with CFO (blue line). For the baseline up-chirp with CFO in the first window with time offset τ, it results in a peak at position $f_1 = f_s + f_{cfo} - k\tau$. For the baseline down-chirp with CFO in the second window, it results in a peak at position $f_2 = f_s + f_{cfo} + k\tau$. We can see the time offset leads to the opposite shift to the peak frequency shift centered at CFO as $f_s = 0$. Therefore, CFO can be calculated as $f_{cfo} = \frac{f_1+f_2}{2}$ and the time offset can be calculated as $\tau = \frac{f_2-f_1}{2k}$.

With Collision Then, we consider the case with collision. Actually, the above method has already been used in other papers (e.g., NScale [15]) to calculate CFO when collision happens. However, They do not consider the situation of SFD collision. When collision happens, there is more than one peak in the demodulation window of SFD as there can be more than one SFD collided together. The existing methods have not given a method to pick the right peak of SFD that corresponding to the preamble of the same packet.

The key challenge here is that there are multiple peaks in each window, and we need to find the preamble and SFD corresponding to the same packet to calculate CFO. Suppose there is an N-packet collision, we first find N preambles by finding N_p peaks at the same frequency. For each preamble, we can get the rough start position and the corresponding frequency f_1. And we need to get the frequency f_2 of the corresponding SFD to calculate the f_{cfo} and τ.

However, as shown in Figure 8, given the peak P of the preamble of the packet 2 with frequency f_1 in w_1, there are more than one SFDs collided together in w_2 when collision happens. Suppose the rough start position of the preamble of the packet 2 is P_{pre_2}, and thus the rough start position of the SFD of the packet 2 is $P_{SFD_2} = P_{pre_2} + T_{chirp} * (N_p + 2)$, as the preamble contains N_p baseline up-chirps and two sync word. The w_1 starts at P_{pre_2}, and the w_2 starts at P_{SFD_2}. Suppose the number of collided SFD is m $(1 \leq m \leq N)$, there are m peaks of these SFDs mixed together, leading to difficulty in getting f_2.

We find the right peak of the SFD of the packet 2 as follows. For each peak p_i with frequency f_i in the m peaks, we calculate the CFO $f_{cfo_i} = \frac{f_1+f_i}{2}$ and $\tau_i = \frac{f_i-f_1}{2k}$. We then check whether p_i is the peak of SFD in the packet 2 based on the f_{cfo_i} and τ_i as shown in Figure 9. We shift the moving window w_2 in Figure 8 by the time offset τ_i and

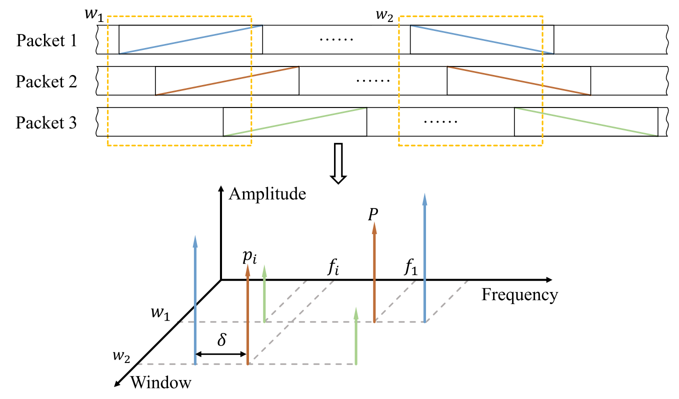

Fig. 8. Peaks for preambles and SFDs in a collision. When there are multiple preambles collided together, there are multiple peaks corresponding to the multiple SFDs in a demodulation window.

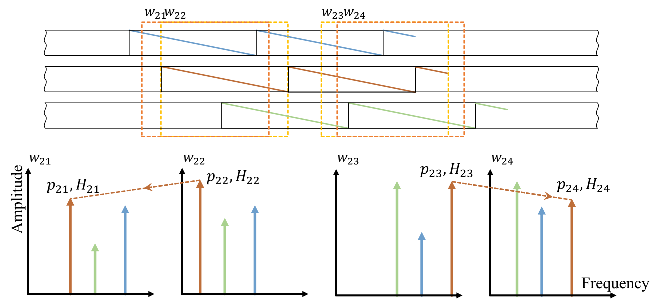

Fig. 9. Calculate CFO in packet collisions.

$\tau_i + 2.25 * T_{chirp}$ to w_{22} and w_{23} in Figure 9. If p_i is the right peak, w_{22} should be aligned with the first baseline down-chirp in the SFD of the packet 2 and w_{23} should terminate at the end of the SFD and contain an entire chirp. Suppose the peaks p_{22} and p_{23} are the corresponding peaks of p_i in w_{22} and w_{23}. H_{22} and H_{23} are their height, respectively. Then we slightly shift the moving window by δ to the left (w_{21}) and right (w_{24}), and calculate the corresponding peak p_{21}, p_{24} and their height H_{21} and H_{24}. p_i is the right peak *iff* $H_{22} > H_{21}$ and $H_{23} > H_{24}$. Otherwise, p_i is not the peak of the SFD of the packet 2.

The value of δ should be carefully set. We need to guarantee that only the SFD of the packet 2 can be aligned with the moving window, i.e., δ should be smaller than the minimum time offset among the m collided SFDs. This is because the frequency of a peak for an SFD is determined by the time offset between the SFD and a window. Therefore, in the same moving window, the frequency offset of peaks among different SFDs is determined by the time offset among those SFDs. Thus, the minimum time offset of SFDs can be calculated by the minimum frequency offset of peaks. As shown in Figure 8, assume the frequency of the closest peak to p_i is $f_{closest}$. The shift δ should be smaller than $abs(f_{closest} - f_i)/k$ where f_i is the frequency of p_i and k is the frequency changing rate of a chirp. Based on δ, we can guarantee that there is no other aligned SFD while shifting the window by δ.

Note that many methods have been proposed to calculate the CFO and the time offset with great performance [20], [24], [25]. However, the models build by these methods apply only to the case of single LoRa packet, and are not applicable in the case of packet collisions. We make full use of the structure of LoRa preamble and SFD and can calculate the CFO and the time offset under collisions.

E. Aligned Chirp Detection

After detecting the exact start of each packet in the received signal, the positions of all chirps are known. We use a collision moving window to align all chirps in the received signal. In each window, we extract all peaks after de-chirp and FFT. Then, we identify the peak of the chirp aligned with the window. When there is no collision, the only peak in the window is the peak of the chirp aligned with the window.

Then, we consider the case with collision. Given a window w_i aligned with chirp c_i in the payload of packet A, assume w_{i-1} and w_{i+1} are the preceding and following window in the moving window sequence. We extract all peaks in those three windows. There should be $2N-1$ peaks for N collided packets in each window.

We first group peaks belonging to the same chirp based on the frequency constraint. For all groups of peaks, we need to identify the peak corresponding to the aligned chirp (i.e., c_i) based on the height constraint. For example, assume p_{i-1}, p_i and p_{i+1} are three peaks in the same group and H_{i-1}, H_i and H_{i+1} are their height, respectively. The peak p_i corresponds to the aligned chirp c_i *iff* the height constraint is satisfied, i.e., $H_{i-1} \leq H_i$ and $H_i \geq H_{i+1}$.

Height Revision. In practice, the height of the peak is influenced by various factors and sometimes cannot reach its highest even aligned with the demodulation window. To prevent errors in height comparison caused by these factors, we first calculate the SNR and get the influence of noise on peak height according to the SNR. Then we give a certain error tolerance during height comparison. For example, assume the influence of noise on peak height is H. We revise the height of these peaks to $H_{i-1} - H$, $H_i + H$, and $H_{i+1} - H$ and then comparing the revised height.

Concurrent Transmission. This aligned chirp detection method mainly works for collisions with a time offset among packets. In practice, the probability for concurrent transmission with no time offset should be very low. Thus, our method can address most of the cases in practice. Even in the case of concurrent transmission, we can extend our method by leveraging existing techniques. First, the height of the peak is influenced by the signal strength and the length of chirp segment in a window. The height of the entire chirps should be similar for the same packet, and should be different for different packets. Thus, we can compare the height of peaks of different packets and divide these peaks to different groups. Second, as introduced in Choir, there is a small frequency distortion due to hardware imperfection. Thus, we can divide these peaks according to the fractional part of the frequency as mentioned in Choir [26]. Thus, we can divide those peaks according to the height of the peak or the fractional part of the frequency distortion and then separate into different packets.

We start decoding from the first window to the last window in the collision moving window sequence based on the above process. For the first (last) window in the sequence, there is no preceding (following) window. Thus, we should find the aligned chirp only based on two windows.

F. Packet Recovery

Till now, we have detected 1) the aligned chirp in each window and 2) the exact start position of each packet. Then, we can group chirps into different packets, decode these packets and identify the LoRa network of each packet.

Packet Separation. According to LoRa physical layer structure and the position of chirp, we can group chirps into the same packet by

$$P_{chirp} = P_{start} + (n + 0.25) * T_{chirp} \tag{10}$$

where P_{chirp} and P_{start} means the start position of n-th chirp signal and LoRa packet in time domain. And we can obtain the length of different collided packets. When there is no peak satisfying the frequency and height constraint, it means there is no chirp aligned with the window. Thus, the corresponding collided packet ends in the last window aligned with the packet. And we can obtain the length of this packet by calculating the position of the last window.

CFO Elimination CFO brings the frequency offset to the chirp, and there is a frequency offset between the demodulated chirp signal and the actual modulated chirp signal. For chirps belong to the same packet, we first correct the demodulated chirp by compensating the frequency offset caused by CFO and then decode the packet. Assume the demodulated frequency for a chirp $f_s{}^*$, then the real modulated data after removing CFO (calculated in Start Position Detection step) is

$$f_s = (f_s{}^* - f_{cfo}) \mod 2^{SF} \tag{11}$$

Network Identify For each demodulated packet, we use the sync word in the preamble to identify the LoRa transmitter it comes from. And we can communicate with the transmitter successfully as the traditional LoRa network.

V. Detail Emphasis

A. Consecutive Identical Chirps

In addition to preamble, there may also be multiple consecutive identical chirps in a payload. These consecutive identical chirps may influence the calculation of the number of collided chirps as these chirps will also appear in the same frequency after de-chirp and FFT in a non-overlapped moving window. And when the position of the moving window is after the start of these consecutive identical chirps (whether in a preamble or a payload), there will be a least two peaks satisfying the frequency and height constraint. One is the peak of the aligned chirp, and the other is the peak of multiple consecutive identical chirps. AlignTrack proposes a Consecutive-Identical-Chirp-Correct (CICC) method to solve the above problems.

Packets Number Calculation: We find the LoRa preamble [17], [18] by finding N_p (the number of baseline up-chirp in a preamble) consecutive peaks at the same frequency. However, when there is a LoRa payload consisting more than N_p consecutive identical chirps, there will also be N_p

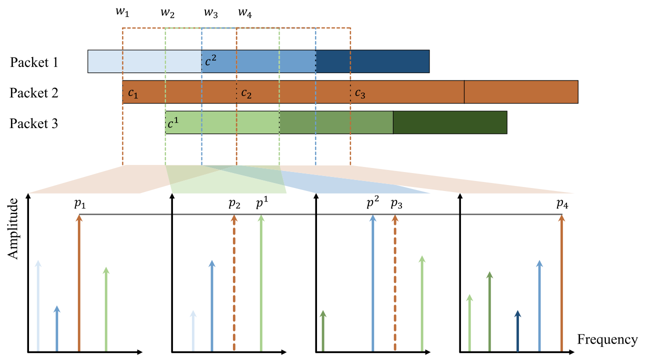

Fig. 10. Aligned chirp detection when there are multiple consecutive identical chirps in a LoRa payload.

consecutive peaks at the same frequency. And thus the number of collided packets will be larger. We suppose the coarse start position of this payload is P_{pl}, and the length of a preamble is L_{pb}. The reason for miscalculating the number of collided packets is we mistake the payload as a preamble. We can remove the wrong calculation about payload in the CFO calculation step. Note that all the positions that contains N_p consecutive peaks are regarded as the preamble positions. And we believe the position of SFD must be behind the preamble positions as the LoRa physical structure shows. When calculating CFO, we first need to estimate the position of SFD $P_{pl} + L_{pb}$ according to the coarse start position of preamble P_{pl}. And then we need to pick out the peak of SFD corresponding to the same packet. However, because there is no SFD in the calculated position $P_{pl} + Lpb$, we cannot get any peak that satisfying the height constraint when shifting the moving window to the left and the right. And thus we can know the signal in P_{pl} is not a preamble and can remove it from the preamble candidates to get the right number of collided packets.

Aligned Chirp Detection: There will be at least two peaks satisfying the frequency and height constraint when there are multiple consecutive identical chirps in a LoRa payload. One is the peak P_a of the aligned chirp, and the other is the peak P_m of multiple consecutive identical chirps. However, we have already find the peak P_m in the last window. And thus we can remove P_m in the current window and correctly find the peak of the aligned chirp.

For example, as shown in Figure 10, there are three consecutive identical chirps in packet 2 and the peaks of these chirps have the same frequency and height from w_1 to w_4, which means that these four peaks can always satisfy the frequency constraint and height constraint. However, p_2 and p_3 are not the peaks corresponding to the aligned chirps in w_2 and w_3. In w_2 and w_3, there are two peaks can be considered as the peak to be detected. And we need to remove the wrong one. When the moving window aligns with the consecutive identical chirps at the first time (w_1 aligns with c_1), there will be only one peak p_1 satisfying the frequency and height constraint and we can get the frequency of p_1. Then we move the moving window to align with c^1 in packet 3, there are two peaks p_2, p^1 satisfying the frequency and height constraint. As we know

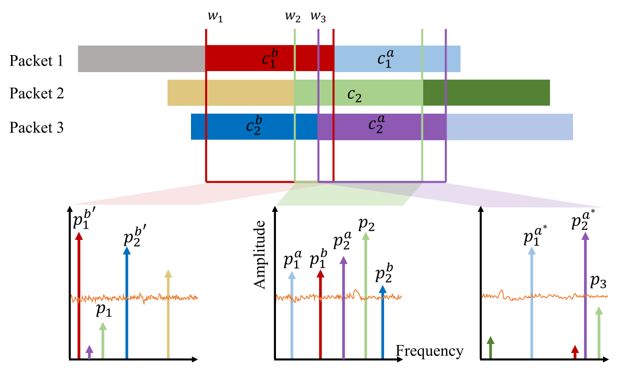

Fig. 11. Aligned chirp detection when peaks of chirp segments cannot be detected: we can remove the peaks of unaligned chirps to derive the right one.

the frequency of p_1 and the time offset between w_1 and w_2, we can calculate the frequency of peak corresponding to the same chirps as p_1 in w_2. And thus we can remove p_2 in w_2 and can find the correct peak p^1 corresponding to c^1. The same method can also be used in the following windows (w_3) when the moving window contains the consecutive identical chirps but not aligns with these chirps.

B. Extremely Low SNR

Ideally, we assume that peaks of segments in the three consecutive windows w_{i-1}, w_i, w_{i+1} should be higher than noise. However, when the SNR is low, or the length of chirp segments in w_{i-1} or w_{i+1} is short, the peak height can be lower than the noise. In this case, existing approaches such as CoLoRa and NScale cannot work as they require to identify all peaks. AlignTrack can deal with this case as follows. For an N-packet collision with chirp c_i aligned with w_i, assume we can extract p_i but cannot extract p_{i-1} and p_{i+1} in w_{i-1} and w_{i+1}, as p_{i-1} and p_{i+1} corresponds to a segment of c_i. There should be $2N-1$ chirp segments in w_i. The peak p_i corresponds to c_i, and other $2N-2$ peaks correspond to segments of the other $2N-2$ chirps, of which $N-1$ chirps c_l^b $(1 \leq l \leq N-1)$ are before c_i and $N-1$ chirps c_l^a $(1 \leq l \leq N-1)$ are after c_i. Assume $p_l^{b'}$ and p_l^b $(1 \leq l \leq N-1)$ are peaks of c_l^b in w_{i-1} and w_i, and $H_l^{b'}$ and H_l^b are their height. As the length of the chirp segment of c_l^b in w_{i-1} must be longer than that in w_i, we have $H_l^{b'} > H_l^b$. If p_l^b can be extracted in w_i, $p_l^{b'}$ can be extracted in w_{i-1}. However, $H_l^{b'}$ and H_l^b do not satisfy the height constraint. Thus, p_l^b are not the peak corresponding to the aligned chirp c_i and can be removed. Similarly, We can remove peaks of c_l^a. After removing peaks of those unaligned chirps, the remaining peak is one corresponding to aligned chirp c_i.

Example. Figure 11 shows a 3-packet collision, and w_1,w_2 and w_3 are three windows aligned with three chirps. Chirp c_2 is aligned with w_2 and we can derive p_2 from w_2. p_1, p_2 and p_3 are peaks of c_2 in w_1, w_2 and w_3. Suppose p_1 and p_3 can not be extracted in w_1 and w_3 due to the impact of noise. In w_2, there are five chirp segments from three LoRa packets and assume we can extract all those five peaks, where p_2 corresponds to c_2, p_1^b and p_2^b correspond to c_1^b from packet 1 and c_2^b from packet 3, p_1^a and p_2^a correspond to c_1^a from packet 1 and c_2^a from packet 3. c_1^b and c_2^b are before c_2 while c_1^a and c_2^a are after c_2.

$p_1^{b'}$ and p_1^b are peaks corresponding to c_1^b in w_1 and w_2. $H_1^{b'}$ and H_1^b denote their height. The segment of c_1^b in w_1 is longer than that in w_2, i.e.,$H_1^{b'} > H_1^b$. p_1^b can be extracted in w_2. Thus, $p_1^{b'}$ can be extracted in w_1. However, $H_1^{b'}$ and H_1^b do not satisfy the height constraint. Thus, we can identify p_1^b as a peak for unaligned chirp and then remove it. Similarly, p_2^b, p_1^a, and p_2^a can be removed.

p_1^a and $p_1^{a^*}$ are peaks corresponding to c_1^a in w_2 and w_3. $p_1^{a^*}$ can also be extracted in w_3. The height of p_1^a and $p_1^{a^*}$ does not satisfy the height constraint. Thus, we can identify p_1^a as a peak for unaligned chirp and then remove it.

Therefore, we can remove all peaks for unaligned chirps from w_2, and the remaining peak is the one for the aligned chirp. Note that in a very low SNR, the peak height of a complete chirp is close to noise amplitude. The peak height of any chirp segment (i.e., a portion of a chirp) can be lower than noise. Thus, only p_i corresponding to c_i in w_i can be extracted. AlignTrack can work in this scenario by removing unaligned chirps. Thus it can work at the same SNR as that of LoRa.

VI. Implementation and Evaluation

A. Implementation

Hardware: We implement our gateway on the HackRF One platform. The HackRF One can run at a frequency range of 30 MHz-6 GHz, and supports the use of GNU-Radio. The maximum sampling rate is 20 M samples per second (Msps), and we only use 1 Msps due to the limited bandwidth. We use commercial LoRa end notes each with an SX1278 chip. In order to better control the LoRa nodes, we also implement the function of LoRa nodes on HackRF one. Therefore, we can use GNURadio+HackRF One as a LoRa transceiver to create packet collisions. HY samples. We implement the LoRa encoder and decoder using Matlab. In our experiment, each LoRa packet consists of a preamble with 10 up-chirps, of which eight are baseline up-chirps, an SFD with 2.25 baseline down-chirps, and a payload with 36 up-chirps. We use $SF = 12$ and $BW = 125$ kHz. The sampling rate is set to 1 MHz, and the central frequency is set to 471.3 MHz.

Software: The HackRF One can provide PHY layer samples of the received signal. We implement AlignTrack in MATLAB on a PC to process LoRa PD with 2.25 baseline down-chirps, and a payload with 36 up-chirps. We use $SF = 12$ and $BW = 125$ kHz in most evaluations. The sampling rate is set to 1 MHz, and the central frequency is set to 471.3 MHz.

Scenario: We use a HackRF One as the receiver, multiple HackRF One nodes as transmitters in the laboratory environment, and LoRa commercial end nodes in the outdoor environment. We evaluate the performance of AlignTrack in two different scenarios. (1) Laboratory environment. We deploy a HackRF One as the receiver, multiple (1-12) HackRF Ones as transmitters within a laboratory environment. We connect all the transmitters to the same laptop through the USB interface (as shown in Figure 12) and use the GNURadio to control

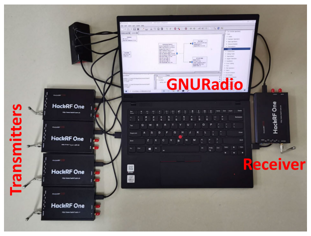

Fig. 12. Experiment setup.

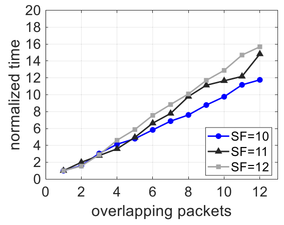

Fig. 13. Normalized time consumption under different number of overlapping packets.

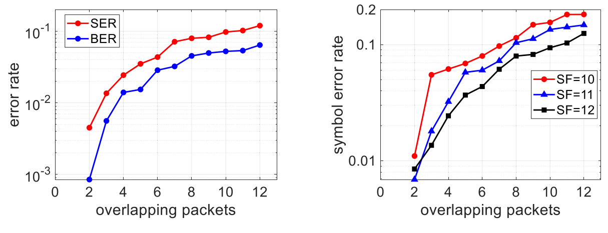

Fig. 14. SER and BER under different number of overlapping packets: (a) SER and BER when $SF = 12$. (b) SER under different SFs.

the transmit time of transmitters to control the number of collided packets. (2) Outdoor real LoRa network. As shown in Figure 21, we evaluate the performance of AlignTrack in two real outdoor LoRa networks: a park of 127m $\times$ 100m with many cars and a playground of 230m $\times$ 140m with many people. We place 12 COTS LoRa end nodes with chip SX1278 as transmitters in different places and use a HackRF One as the receiver. we control the number of collided packets by controlling the number of nodes in the transmitting state in the network.

Note that AlignTrack does not depend on the specific hardware platform. It can decode collided packets as long as the PHY samples of a received signal are provided. AlignTrack sits at the gateway for collision decoding. It can work for existing COTS LoRa nodes without any modification in software and hardware.

B. Evaluation

We mainly measure the following metrics: (1) symbol error rate (SER), the error rate of decoding a chirp, (2) bit error rate (BER), the error rate after translating symbols to bits, and (3) throughput, the total receiving symbol rate (symbols/second) at the receiver. Note that, in LoRa the BER is usually lower than SER as it uses error correcting codes at the symbol level.

Based on those metrics, we mainly evaluate the performance of AlignTrack under (1) different number of overlapping packets, (2) different SNR, (6) different time offset, and (4) different spreading factors among packets. Further, we compare AlignTrack with the mLoRa, FTrack, CoLoRa, NScale and Choir. Packets sent by each transmitter are known in advance to calculate the SER, BER, and throughput. In order to show the collision decoding performance in real LoRa environments, our experiments are conducted in the scenario of $SNR < 0$ dB and the collision can appear at different positions, such as preamble-preamble, preamble-SFD, preamble-payload, etc.

1) Time Consumption: AlignTrack adopts a moving window to align with all chirps in the received signal, which means the time consumption is influenced by the number of chirps. We create different SFs and number of overlapping packets using MATLAB. Each packet consists of 36 chirps in the payload. The total number of chirp to be decoded is linearly related to the number of overlapping packets, which means the time consumption should also be linearly related to the number of overlapping packets. Figure 13 shows the normalized time consumption of AlignTrack under different SF and number of overlapping packets. The normalized time consumption increases when the number of overlapping packets increases. The normalized time consumption of decoding 12 packets is 12$\times$ and 15$\times$ of decoding 1 packet when $SF = 10$ and $SF = 12$. The time consumption increases faster when the SF increases. This is due to the reason that when the SF increases, the chirp length increases, and it will spend more time on *Peak Extraction*. In the future, we will further work on how to reduce time consumption.

2) Impact of the Number of Packets: We evaluate the performance of AlignTrack at the different number of overlapping packets in a laboratory environment. We use a HackRF One as the receiver and use 12 HackRF Ones as transmitters. We use GNURadio to control the transmit time of each transmitter to create collisions.

Figure 14(a) shows the averaged SER and BER with the different number of overlapping packets when $SF = 12$. We can see that the overall SER is under 4% and BER is under 2% when the overlapping number is under 6, the SER is under 10% and the BER is under 6% when the overlapping number is under 10. The maximum number of overlapping packets is 12 with BER $< 6.5\%$, which is acceptable in the LoRa network. Both SER and BER increase as the overlapping number increases from 1 to 12. It is because the time offset among packets decreases when the overlapping number increases. A smaller time offset leads to less height change among different windows and thus degrades the decoding performance. Nevertheless, most existing approaches such as FTrack, CoLoRa, and NScale cannot work well under a small time offset.

Figure 14(b) shows the averaged SER with difference SFs in different number of overlapping packets. We can see that the overall SER is under 10% when the overlapping number is under 7, the SER is under 20% when the overlapping number is 12 for all three SFs. The SER increases with the increase of overlapping number and the SER is lower with higher SF.

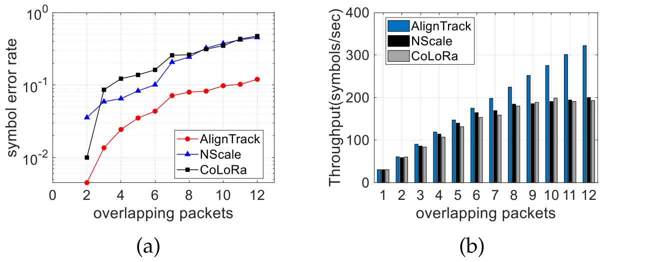

Fig. 15. SER and Throughput comparison in different number of overlapping packets: (a) Averaged SER. (b) Network Throughput.

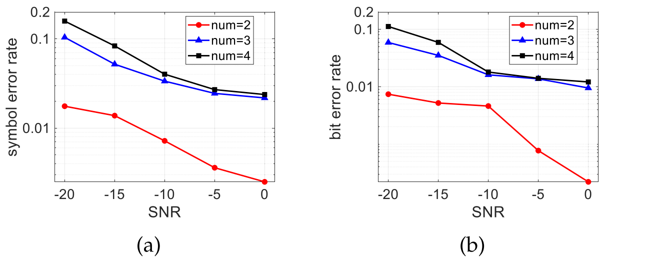

Fig. 16. SER and BER under different SNR: (a) SER under different SNR. (b) BER under different SNR.

This is because the symbol length increases with the SF increases and the longer symbol can concentrates more power to resist interference.

3) Comparison With State-of-the-Arts Under Different Numbers: We compare AlignTrack with NScale and CoLoRa. Figure 15 shows the averaged SER and throughput when the number of overlapping packets varies from 1 to 12. Figure 15(a) shows the averaged SER. The SER increases for all three methods with the increasing number of overlapping packets. The SER of AlignTrack is much lower than that of NScale and CoLoRa. The SER of CoLoRa and NScale is more than 40%, while that of AlignTrack is still lower than 12% in a 12-packet collision. This is because CoLoRa and NScale partition a chirp to segments and use the height difference among packets to decode collisions. NScale uses a non-stationary scaled down-chirp which should be carefully designed. When the number of overlapping packets increases, the height difference among chirps in different LoRa packets decreases, and the features of different packets are more likely to be similar, i.e., not distinguishable enough to separate packets. For AlignTrack, we only focus on the peak frequency and height change of the aligned chirp at the current moving window. Thus, our approach introduces more robust features to distinguish packets.

Figure 15(b) shows the network throughput. The network throughput increases, and the throughput of AlignTrack is always higher than that of NScale and CoLoRa. When the number of overlapping packets reaches 12, the throughput of AlignTrack is 322 sps, which is 1.61× of NScale (200sps) and 1.68× of CoLoRa (192sps).

4) Impact of SNR: We evaluate the impact of the signal-to-noise ratio (SNR) on the performance of AlignTrack. Due to the collisions of multiple packets, the actual interference intensity is higher than the ambient noise intensity. We use $3 \sim 5$ HackRF Ones, of which one is the receiver and the others are transmitters. To accurately control the SNR, we add additive white Gaussian noise (AWGN) to the received signal.

AlignTrack multiplies the entire up-chirp with baseline down-chirp in each moving window, which is the same as traditional LoRa. Figure 16 shows the averaged SER and BER of AlignTrack when SNR varies from -20 to 0. The SER and BER decrease with the increase of SNR. When $SNR \approx -20$ dB, the SER of 2-packet collision is 1.76%, and the BER is 0.74%, i.e., AlignTrack can decode most 2-packets collision successfully at an extremely low SNR. When $SNR = 0$ dB, the SER and BER of 2-packet collision

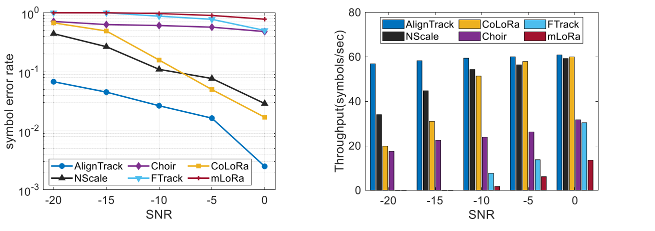

Fig. 17. SER and throughput comparison under different SNR: (a) Averaged SER. (b) Network Throughput.

are 0.25% and 0.022%. For 4-packet collisions, the SER is still lower than 10%, and BER is lower than 6% even at $SNR \approx -15$ dB. In all, AlignTrack can decode packets collision in an extremely low SNR as traditional LoRa.

5) Comparison With State-of-the-Arts Under Different SNR: Figure 17 shows averaged SER and throughput of different methods with the SNR varies from -20 to 0. We compare AlignTrack with mLoRa, FTrack, CoLoRa, NScale, and Choir. Figure 17(a) shows that the SER of AlignTrack is much lower than that of the other methods. When $SNR \approx 0$ dB, the SER of mLoRa reaches 77.8%, and the SER of FTrack and Choir is about 50%, while that of AlignTrack is even lower than 0.3%. When $SNR \approx -20$ dB, the SER of AlignTrack is still lower than 7%, while the SER of NScale is 44.25% and the SER of CoLoRa and Choir is about 70%. This is because AlignTrack transforms time domain information to frequency domain information and uses the entire up-chirp to demodulate. AlignTrack can concentrate the energy of the entire chirp to resist interference from other packets and noise. mLoRa and FTrack only use time-domain information and cannot work at a low SNR. The hardware offset in Choir is also hard to find in a low SNR. CoLoRa and NScale separate the entire chirp into two segments, which reduces the concentration of energy. Thus, AlignTrack introduces much less SNR loss than other approaches that are using parts of a chirp.

Figure 17(b) shows the network throughput. When $SNR \approx -20$ dB, the network throughput of AlignTrack is 57 sps, which is 1.68× of NScale (34sps), 3.35× of Choir (17sps) and 3× of CoLoRa (19sps).

6) Impact of Symbol Time Offset: The height of a peak is impacted by chirp segment length in the current window. A small symbol time offset among packets leads to a small difference of segment length between two windows. This leads

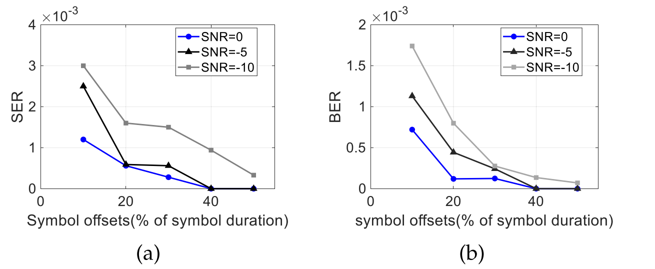

Fig. 18. SER and BER under different symbol time offsets in collisions: (a) SER. (b) BER.

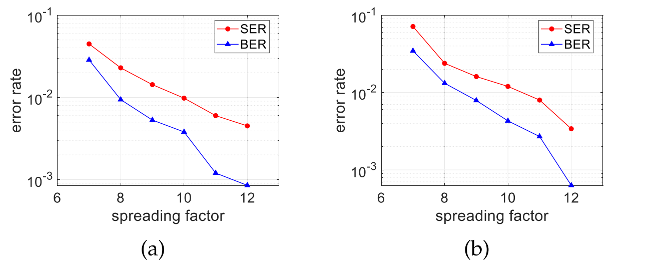

Fig. 19. SER and BER under different SFs:(a) SNR = 0. (b) SNR = −5.

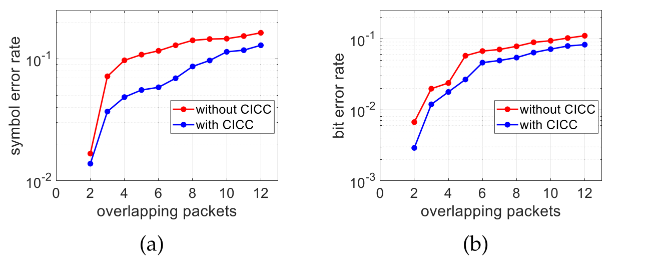

Fig. 20. SER and BER with/without CICC method: (a) SER. (b) BER.

to a very small height change of a peak. Thus, we evaluate how symbol time offset can impact the performance of AlignTrack.

We receive non-collided packets from two transmitters respectively and add a segment of noise with different lengths before the start of these packets to create signals of which the start of the LoRa signal is different. We then mix the signals from different transmitters in pairs to create many 2-packet collisions with different symbol time offsets. Figure 18 shows the averaged SER and BER under the impact of symbol time offset and SNR. The SER and BER decrease with the increase of the symbol time offset, which coincides with our analysis. When the time offset is larger, it is easier for AlignTrack to find the unique peak corresponding to the aligned chirp. The smallest symbol time offset is 10% of symbol duration when $SER < 0.3\%$, $BER < 0.2\%$ and $SNR = -10$ dB. This means that AlignTrack can decode almost all overlapping packets under a very small time offset. The collision packets that cannot be decoded when the time offset is too small can be retransmitted.

7) Impact of Spreading Factor: In this experiment, we evaluate the impact of the spreading factor(SF) on the performance of AlignTrack. Figure 19 shows the averaged SER and BER of AlignTrack when SF varies from 7 to 12 under different SNR. Both the SER and BER decrease with the increase of SF. When $SF = 7$, the BER of 2-packet collision is 2.86% and 3.46% when $SNR = 0$ and $SNR = -5$, i.e., AlignTrack can decode most 2-packet collision successfully at a low SF. When $SNR = -5$, the SER is lower than 2% and the BER is lower than 1% when $SF > 8$. The BER is lower than 1% when $SF > 7$ and $SNR = 0$. In all, this experiment indicates that AlignTrack can work in all SFs that can be used in LoRa.

8) Impact of Consecutive Identical Chirp: As mentioned in Section V-A, the consecutive identical chirps can bring errors in the Preamble Detection and the Aligned Chirp Detection module in the overall demodulation process of AlignTrack in Section IV. And AlignTrack propose a Consecutive-Identical-Chirp-Correct method (CICC) to estimate the influence of the consecutive identical chirps. We evaluate the performance of the CICC under different number of overlapping packets. The packets sent by each transmit contains a payload with 36 up-chirps, of which there are three consecutive identical up-chirps.

Figure 20 shows the averaged SER and BER when demodulating the collided packets with/without the CICC. Figure 20 shows that the SER and BER with the CICC are always lower than that without CICC. When the number of overlapping packets is larger than 5, the SER without CICC is larger than 10% while the SER with CICC is lower than 6%. When the number of overlapping packets is 12, the BER without CICC is 11.04% and the BER with CICC is still lower than 10% (8.26%) when the number of overlapping packets is 12, which is acceptable in LoRa demodulation. Therefore, CICC can help find out the right aligned chirp and improve the accuracy of the collided-packet demodulation when there are consecutive identical chirps collided together.

9) Performance in an Outdoor Network: We evaluate the performance of AlignTrack in two real outdoor LoRa networks: a park and a playground. We use COTS LoRa nodes with chip SX1278 as transmitters and a HackRF One as the receiver. As shown in Figure 21, we deploy LoRa nodes in 12 different places.

Figure 22 shows the averaged SER under different outdoor networks. The SER increases with the increasing number of overlapping packets. The overall SER is under 10% when the overlapping number is under 7, the SER is under 15% when the overlapping number is 12 in these two networks. The increase of SER shows the same pattern in a playground and in a park. This indicates the capability of AlignTrack to resist to environment noise and achieve stable performances in different environments. However, the SER in a park in usually larger than the SER in a playground. This is because the environment of a park is more complex than that of a playground.

Figure 23 shows the averaged SER and throughput of four methods when the number of overlapping packets varies from 1 to 12. Figure 23(a) shows the averaged SER. The SER of NScale, CoLoRa, and Choir increase faster than that of AlignTrack, and the SER of Choir increases the fastest. This is because Choir uses hardware imperfection which is difficult to find under low SNR, and NScale and CoLoRa partition a chirp to segments and use the height difference among packets to decode collisions. When the number of overlapping

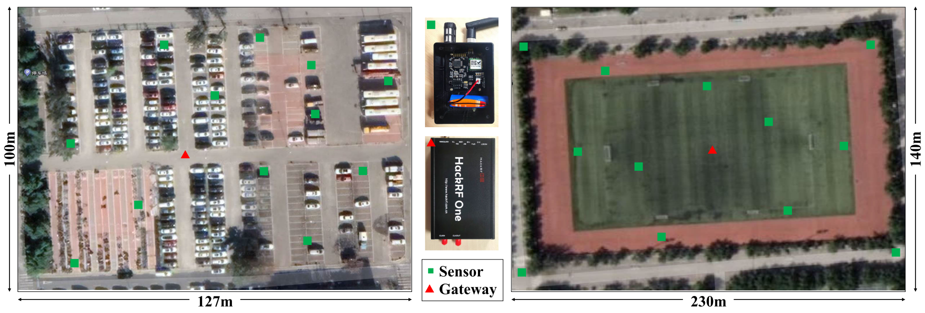

Fig. 21. Outdoor LoRa Network: LoRa nodes with radio chip SX1278 as transmitters and HackRF One as the receiver (gateway) in a park and a playground.

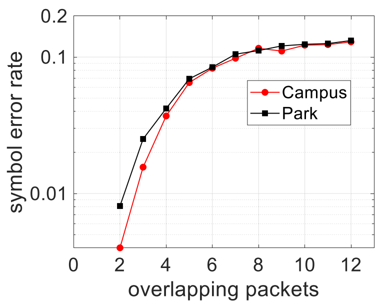

Fig. 22. SER under different outdoor environments.

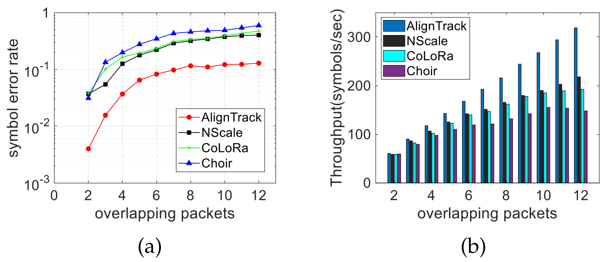

Fig. 23. Performance in real outdoor deployed LoRa networks:(a) Averaged SER. (b) Network throughput.

packets reaches 7, the SER of Choir is more than 40%, the SER of NScale and CoLoRa are almost 30% while the SER of AlignTrack is still lower than 10%. When the number of overlapping packets reaches 12, the SER of NScale is more than 40%, which is 3.1× of AlignTrack (12.9%). And the SER of CoLoRa (48%) and Choir (60%) are 3.72× and 4.65× of AlignTrack. Figure 23(b) shows the network throughput of AlignTrack, NScale, CoLoRa and Choir. When the number of overlapping packets reaches 12, the throughput of AlignTrack is 320 sps, which is 1.52× of NScale (210 sps), 1.68× of CoLoRa (190 sps), and 2.16× of Choir (148 sps).

VII. Related Work

Collision resolution in LoRa. There are many works to decode collisions for LoRa. mLoRa [12] adopts SIC to separate packets and can decode collision from three nodes. FTrack [13] separates collided packets by calculating the continuity of instantaneous frequency and concentrates on the time domain features. Choir [26] exploits the frequency shift of imperfect hardware to separate packets. CoLoRa [14] considers the height relationship of peaks in adjacent windows. NScale [15] multiples the chirp segment with a non-stationary scaled down-chirp and leverages peaks of different segments to distinguish different packets. SCLoRa [27] leverages multidimensional information and utilizes cumulative spectral coefficient to separate symbols in the overlapped signal. Pyramid [28] utilizes a sliding window to translate the time offsets of collided chirps to frequency and power feature for chirp decoding. PCube [29] designs a phase-based parallel decoder that can scale the concurrent transmissions of LoRa nodes with reception diversities of multiple receiving antennas of a gateway. CIC [30] adopts a spectral intersection operation to demodulate symbols via canceling out all interfering symbols. AlignTrack leverages the entire chirp in LoRa and introduces a very small SNR loss and does not need to do any redundant operations.

General methods for collision resolution. There are many traditional methods for collision resolution. One is to use multiple antennas, such as Multi Input Multi Output(MIMO) [31], [32]. MIMO uses multiple antennas to form a transceiver system with multiple channels. However, MIMO is not suitable for a single antenna device like the LoRa node. The other is to perform channel detection to avoid collision, such as CSMA/CA [33], [34] and RTS-CTS [35], [36]. However, CSMA/CA needs to detect the channel status, and RSSI based channel detect method does not work in a very low SNR like the LoRa network. The impact of the hidden/exposed terminal problems is exacerbated and will significantly reduce the network efficiency. Channel Activity Detection (CAD) method introduced by LoRa needs to detect the LoRa preamble, which introduces a high overhead and power consumption. SIC [37] iteratively finds and reconstructs coding information based on different signal strengths and some known coding information. However, it does not utilize the features of LoRa.

Improvement on IoTs. A variety of works have been proposed to improve the performance of IoTs. OPR [38] uses multiple gateways to recover packets subject to CRC and/or FEC errors. Chime [39] analyzes the path signals traverse from the client to distributed and coordinated gateways to choose the optimal frequency of the LPWAN client. LiteNap [40] enables sub-Nyquist sampling and packet decoding to improve energy efficiency. EF-LoRa [41] allocates different network resources to achieve fair energy consumption among end devices. The work [42] conducts a series of experiments to verify the claims made by Semtech. L2X [43] provides long-range CTC to diverse receivers with LoRa transmitters. NELoRa [44] exploits the feature abstraction ability of deep learning to support ultra-low SNR LoRa communication. Falcon [45] addresses the link dynamics by enabling data transmission for every low SNR or even disconnected LoRa links. P^2LoRa [46] is the first ambient LoRa backscatter system with parallel decoding and long-range communication. This work [47] proposes a new energy harvesting mechanism to support continuous communication. Mourade et al. [48] propose an enhanced authentication protocol for IoT. Jiehong et al. [49] proposes a Hierarchical Adaptive Energy-efficient Clustering Routing strategy to reduce and balance network energy consumption.

Analyzes on LoRa. Many methods have been proposed to analyze the whole process of LoRa demodulation. Augustin et al. [16] provides an overview of LoRa and an in-depth analysis of its functional components. Kang et al. [17] thoroughly analyzes the performance of the preamble detection model in the LoRa demodulation process and develops a optimal preamble detection scheme to maximum the detection probability. FlipLoRa [50] encodes packets with interleaved upchirps and downchirps instead of only using upchirps as in LoRa. This work [51] presents a comprehensive understanding of LoRa PHY (physical layer protocol) and reveals the fundamental reasons for the performance gap. Ameloot et al. [21] presents an independently developed packet reception algorithm, which drastically improves the physical performance of LoRa communication links. Edward et al. [22] investigates different preamble detection schemes to refrain from the consecutive constraint. Xhonneux et al. [20] builds a low-complexity, yet efficient synchronization algorithm capable of correcting the sampling time offsets and the carrier frequency offsets. Ostinato [52] enables communication for weak links and enhances the coverage for real deployments of COTS LoRa.

VIII. Conclusion

LoRa has been one of the key technologies to connect millions of devices in the Internet of Things. However, the collision of LoRa packets significantly degrades its performance in practice. Existing collision decoding approaches introduce non-negligible SNR loss in collision decoding. We present AlignTrack, the first LoRa collision decoding approach that incurs negligible SNR loss and can decode collisions under a very low SNR. Unlike state-of-the-arts that leverage partial chirps to separate collided packets, AlignTrack aligns windows to each packet and leverages the aligned chirps in each window to separate packets. We address practical challenges in the implementation. We propose a method to accurately find the start of each packet under interference and CFO. We show how to accurately find the aligned chirps in each window and recover accurate peak information. We implement AlignTrack on the HackRF One platform and evaluate its performance extensively. AlignTrack totally sits at the server side without any modification to the LoRa end nodes and can be applied to existing LoRa networks. The evaluation results show that AlignTrack improves network throughput by 1.68× compared with NScale and 3× compared with CoLoRa.

References

[1] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, "Overview of cellular LPWAN technologies for IoT deployment: Sigfox, LoRaWAN, and NB-IoT," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2018, pp. 197–202.

[2] O. Khutsoane, B. Isong, and A. M. Abu-Mahfouz, "IoT devices and applications based on LoRa/LoRaWAN," in *Proc. 43rd Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2017, pp. 6107–6112.

[3] A. Mdhaffar, T. Chaari, K. Larbi, M. Jmaiel, and B. Freisleben, "IoT-based health monitoring via LoRaWAN," in *Proc. IEEE EUROCON 17th Int. Conf. Smart Technol.*, Jul. 2017, pp. 519–524.

[4] D. Davcev, K. Mitreski, S. Trajkovic, V. Nikolovski, and N. Koteli, "IoT agriculture system based on LoRaWAN," in *Proc. 14th IEEE Int. Workshop Factory Commun. Syst. (WFCS)*, Jun. 2018, pp. 1–4.

[5] R. F. A. M. Nor, F. H. K. Zaman, and S. Mubdi, "Smart traffic light for congestion monitoring using LoRaWAN," in *Proc. IEEE 8th Control Syst. Graduate Res. Colloq. (ICSGRC)*, Aug. 2017, pp. 132–137.

[6] S. A. A'ssri, F. H. K. Zaman, and S. Mubdi, "The efficient parking bay allocation and management system using LoRaWAN," in *Proc. IEEE 8th Control Syst. Graduate Res. Colloq. (ICSGRC)*, Aug. 2017, pp. 127–131.

[7] Y. Peng et al., "PLoRa: A passive long-range data network from ambient LoRa transmissions," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Aug. 2018, pp. 147–160.

[8] V. Talla, M. Hessar, B. Kellogg, A. Najafi, J. R. Smith, and S. Gollakota, "LoRa backscatter: Enabling the vision of ubiquitous connectivity," *Proc. ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 1, no. 3, pp. 1–24, 2017.

[9] N. Abramson, "The ALOHA system: Another alternative for computer communications," in *Proc. Fall Joint Comput. Conf.*, 1970, pp. 281–285.

[10] J. D. C. Silva, J. J. Rodrigues, A. M. Alberti, P. Solic, and A. L. Aquino, "LoRaWAN—A low power wan protocol for Internet of Things: A review and opportunities," in *Proc. 2nd Int. Multidisciplinary Conf. Comput. Energy Sci. (SpliTech)*, Jul. 2017, pp. 1–6.

[11] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of LoRaWAN," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 34–40, Sep. 2017.

[12] X. Wang, L. Kong, L. He, and G. Chen, "MLoRa: A multi-packet reception protocol in LoRa networks," in *Proc. IEEE 27th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2019, pp. 1–11.

[13] X. Xia, Y. Zheng, and T. Gu, "FTrack: Parallel decoding for LoRa transmissions," in *Proc. 17th Conf. Embedded Networked Sensor Syst.*, Nov. 2019, pp. 192–204.

[14] S. Tong, Z. Xu, and J. Wang, "CoLoRa: Enabling multi-packet reception in LoRa," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Jul. 2020, pp. 2303–2311.

[15] S. Tong, J. Wang, and Y. Liu, "Combating packet collisions using non-stationary signal scaling in LPWANs," in *Proc. 18th Int. Conf. Mobile Syst., Appl., Services*, Jun. 2020, pp. 234–246.

[16] A. Augustin, J. Yi, T. Clausen, and W. Townsley, "A study of LoRa: Long range & low power networks for the Internet of Things," *Sensors*, vol. 16, no. 9, p. 1466, Sep. 2016.

[17] J.-M. Kang, D.-W. Lim, and K.-M. Kang, "On the LoRa modulation for IoT: Optimal preamble detection and its performance analysis," *IEEE Internet Things J.*, vol. 9, no. 7, pp. 4973–4986, Apr. 2022.

[18] N. E. Rachkidy, A. Guitton, and M. Kaneko, "Decoding superposed LoRa signals," in *Proc. IEEE 43rd Conf. Local Comput. Netw. (LCN)*, Oct. 2018, pp. 184–190.

[19] Spectral leakage. (2015). [Online]. Available: https://en.wikipedia.org/wiki/Spectral_leakage

[20] M. Xhonneux, O. Afisiadis, D. Bol, and J. Louveaux, "A low-complexity LoRa synchronization algorithm robust to sampling time offsets," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3756–3769, Mar. 2022.

[21] T. Ameloot, H. Rogier, M. Moeneclaey, and P. Van Torre, "LoRa signal synchronization and detection at extremely low signal-to-noise ratios," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8869–8882, Jun. 2022.

[22] P. Edward, A. Muhammad, S. Elzeiny, M. Ashour, T. Elshabrawy, and J. Robert, "Enhancing the capture capabilities of LoRa receivers," in *Proc. Int. Conf. Smart Appl., Commun. Netw. (SmartNets)*, Dec. 2019, pp. 1–6.

[23] A. Rahmadhani and F. Kuipers, "When LoRaWAN frames collide," in *Proc. 12th Int. Workshop Wireless Netw. Testbeds, Exp. Eval. Characterization*, Oct. 2018, pp. 89–97.

[24] C. Bernier, F. Dehmas, and N. Deparis, "Low complexity LoRa frame synchronization for ultra-low power software-defined radios," *IEEE Trans. Commun.*, vol. 68, no. 5, pp. 3140–3152, May 2020.

[25] R. Ghanaatian, O. Afisiadis, M. Cotting, and A. Burg, "LoRa digital receiver analysis and implementation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 1498–1502.

[26] R. Eletreby, D. Zhang, S. Kumar, and O. Yağan, "Empowering low-power wide area networks in urban settings," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Aug. 2017, pp. 309–321.

[27] B. Hu, Z. Yin, S. Wang, Z. Xu, and T. He, "SCLoRa: Leveraging multi-dimensionality in decoding collided LoRa transmissions," in *Proc. IEEE 28th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2020.

[28] Z. Xu, P. Xie, and J. Wang, "Pyramid: Real-time LoRa collision decoding with peak tracking," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2021, pp. 1–9.

[29] X. Xia, N. Hou, Y. Zheng, and T. Gu, "PCube: Scaling LoRa concurrent transmissions with reception diversities," in *Proc. 27th Annu. Int. Conf. Mobile Comput. Netw.*, Oct. 2021, pp. 670–683.

[30] M. O. Shahid, M. Philipose, K. Chintalapudi, S. Banerjee, and B. Krishnaswamy, "Concurrent interference cancellation: Decoding multi-packet collisions in LoRa," in *Proc. ACM SIGCOMM Conf.*, Aug. 2021, pp. 503–515.

[31] L. Lu, G. Y. Li, A. L. Swindlehurst, A. Ashikhmin, and R. Zhang, "An overview of massive MIMO: Benefits and challenges," *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 5, pp. 742–758, Oct. 2014.

[32] D. Gesbert, M. Shafi, D. Shiu, P. J. Smith, and A. Naguib, "From theory to practice: An overview of MIMO space-time coded wireless systems," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 3, pp. 281–302, Apr. 2003.

[33] R. Laufer and L. Kleinrock, "The capacity of wireless CSMA/CA networks," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1518–1532, Jun. 2016.

[34] E. Ziouva and T. Antonakopoulos, "CSMA/CA performance under high traffic conditions: Throughput and delay analysis," *Comput. Commun.*, vol. 25, no. 3, pp. 313–321, Feb. 2002.

[35] P. Lin and L. Zhang, "Full-duplex RTS/CTS aided CSMA/CA mechanism for visible light communication network with hidden nodes under saturated traffic," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[36] K. Xu, M. Gerla, and S. Bae, "Effectiveness of RTS/CTS handshake in IEEE 802.11 based ad hoc networks," *Ad Hoc Netw.*, vol. 1, no. 1, pp. 107–123, Jul. 2003.

[37] P. Patel and J. Holtzman, "Analysis of a simple successive interference cancellation scheme in a DS/CDMA system," *IEEE J. Sel. Areas Commun.*, vol. 12, no. 5, pp. 796–807, Jun. 1994.

[38] A. Balanuta, N. Pereira, S. Kumar, and A. Rowe, "A cloud-optimized link layer for low-power wide-area networks," in *Proc. 18th Int. Conf. Mobile Syst., Appl., Services*, Jun. 2020, pp. 247–259.

[39] A. Gadre, R. Narayanan, A. Luong, A. Rowe, B. Iannucci, and S. Kumar, "Frequency configuration for low-power wide-area networks in a heartbeat," in *Proc. 17th USENIX Symp. Networked Syst. Design Implement. (NSDI)*, 2020, pp. 339–352.

[40] X. Xia, Y. Zheng, and T. Gu, "LiteNap: Downclocking LoRa reception," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Jul. 2020, pp. 2632–2645.

[41] W. Gao, W. Du, Z. Zhao, G. Min, and M. Singhal, "Towards energy-fairness in LoRa networks," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 788–798.

[42] J. C. Liando, A. Gamage, A. W. Tengourtius, and M. Li, "Known and unknown facts of LoRa: Experiences from a large-scale measurement study," *ACM Trans. Sen. Netw.*, vol. 15, no. 2, pp. 1–35, Feb. 2019.

[43] S. Tong, Y. He, Y. Liu, and J. Wang, "De-spreading over the air: Long-range CTC for diverse receivers with LoRa," in *Proc. 28th Annu. Int. Conf. Mobile Comput. Netw.*, Oct. 2022, pp. 42–54.

[44] C. Li et al., "NELoRa: Towards ultra-low SNR LoRa communication with neural-enhanced demodulation," in *Proc. 19th ACM Conf. Embedded Networked Sensor Syst.*, Nov. 2021, pp. 56–68.

[45] S. Tong, Z. Shen, Y. Liu, and J. Wang, "Combating link dynamics for reliable LoRa connection in urban settings," in *Proc. 27th Annu. Int. Conf. Mobile Comput. Netw.*, Oct. 2021, pp. 642–655.

[46] J. Jiang, Z. Xu, F. Dang, and J. Wang, "Long-range ambient LoRa backscatter with parallel decoding," in *Proc. 27th Annu. Int. Conf. Mobile Comput. Netw.*, Oct. 2021, pp. 684–696.

[47] B. Pavani, L. N. Devi, and K. V. Subbareddy, "Energy enhancement and efficient route selection mechanism using H-SWIPT for multi-hop IoT networks," *Intell. Converged Netw.*, vol. 3, no. 2, pp. 173–189, Jun. 2022.

[48] M. Azrour, J. Mabrouki, A. Guezzaz, and Y. Farhaoui, "New enhanced authentication protocol for Internet of Things," *Big Data Mining Analytics*, vol. 4, no. 1, pp. 1–9, Mar. 2021.

[49] J. Wu, X. Sun, J. Wu, and G. Han, "Routing strategy of reducing energy consumption for underwater data collection," *Intell. Converged Netw.*, vol. 2, no. 3, pp. 163–176, Sep. 2021.

[50] Z. Xu, S. Tong, P. Xie, and J. Wang, "FlipLoRa: Resolving collisions with up-down quasi-orthogonality," in *Proc. 17th Annu. IEEE Int. Conf. Sens., Commun., Netw. (SECON)*, Jun. 2020, pp. 1–9.

[51] Z. Xu, P. Xie, S. Tong, and J. Wang, "From demodulation to decoding: Towards complete LoRa PHY understanding and implementation," *ACM Trans. Sensor Netw.*, Jul. 2022, doi: 10.1145/3546869.

[52] Z. Xu, P. Xie, J. Wang, and Y. Liu, "Ostinato: Combating LoRa weak links in real deployments," in *Proc. IEEE 30th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2022, pp. 1–11.

Qian Chen (Graduate Student Member, IEEE) received the B.E. degree from the School of Software, Tsinghua University, in 2020, where she is currently pursuing the Ph.D. degree. Her research interests include low-power wide-area networks, the Internet of Things, and wireless network localization.

Jiliang Wang (Senior Member, IEEE) received the B.E. degree in computer science and technology from the University of Science and Technology of China and the Ph.D. degree in computer science and engineering from The Hong Kong University of Science and Technology. He is currently an Associate Professor with the School of Software, Tsinghua University. His research interests include wireless networks, the Internet of Things, and mobile computing.