AlignTrack: Push the Limit of LoRa Collision Decoding

Qian Chen, Jiliang Wang
School of Software, Tsinghua University
cq20@mails.tsinghua.edu.cn, jiliangwang@tsinghua.edu.cn

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 in a very low SNR. 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.

I. INTRODUCTION

Recent years have witnessed the rapid development of Internet of Things (IoT) 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]. 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. However, a typical LoRa gateway can only receive LoRa packets from eight channels. While in practice, a gateway is supposed to connect with thousands of nodes or even more. 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 further 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. 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 [14] [15]. Thus, those approaches cannot work in many practical scenarios.

Our design. To push the limit of LoRa collision decoding, we propose AlignTrack, a novel LoRa packet collision decoding approach to minimize SNR loss. AlignTrack can decode collisions with the same SNR requirement of LoRa. LoRa modulates data with chirps of linearly increasing frequency with different start frequency shifts to encode data bits. The
start frequency shift $f_s$ is decoded by de-chirp: a chirp is multiplied with a standard linearly decreasing down-chirp, which leads to a single tone at frequency $f_s$.

Figure 1 shows three collided packets. 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. 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 window to different packets.

Our main idea is to find the peak of the chirp aligned with each 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. We find that the frequency of peaks corresponding to the same chirp is proportional to the time offset 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 $f_s + k\tau_2$, respectively. Meanwhile, the peak of a chirp in different windows reaches its highest height when the chirp is aligned with the window. AlignTrack leverages this to find the peak of the chirp aligned with each window and then separate packets. For example, AlignTrack first finds 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 determines the peak at $f_s$ corresponds to the chirp $c_2$ aligned with window $w_2$. Then, we can group peaks to different packets based on aligned chirps and decode those packets.

**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. We leverage the preamble and SFD in each packet. The preamble contains baseline up-chirps of linearly increasing frequency, while the SFD contains baseline down-chirps of linearly decreasing frequency. CFO introduces frequency shifts to the up-chirp and down-chirp, and we can estimate the CFO by combining up-chirps and down-chirps. Further, we find the above CFO estimation method fails in collided packets due to the challenge of finding SFD and preamble 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 peak height for a low SNR signal is close to the noise. Thus, using a pre-determined height threshold may fail to identify all 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, and iteratively find all peaks.

(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 peaks 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.**

- We propose AlignTrack to push the limit of decoding low SNR LoRa packet collisions. 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.
- We address non-trivial practical challenges such as inter-packet interference and the impact of CFO in the practical design of AlignTrack.
- 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 change. The evaluation results show that AlignTrack improves the network throughput by $1.68\times$ compared with NScale [15] and $3\times$ compared with CoLoRa [14].

## 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. A baseline chirp is represented as $C(t) = e^{j2\pi(f_0 + kf_t)t}$, where $|k| = BW/T_{chirp}$ is the frequency changing rate, $f_0$ 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., $T_{chirp} = 2^{SF}/BW$. A chirp is an up-chirp when $k > 0$, and otherwise is a down-chirp. A chirp is a baseline up-chirp if $f_0 = -BW/2$ with frequency linearly increasing from $-BW/2$ to $BW/2$.

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

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

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

To demodulate $C(t, f_s)$, LoRa first multiplies it with the baseline down-chirp $C^{-1}(t)$ ($C^{-1}(t)$ is the conjugate of the
2.25 baseline down-chirps. The payload contains data bits to determine the start of the LoRa packet and 2 up-chirps with payload. The preamble contains 6 of three parts: preamble, start frequency delimiter (SFD), and B. LoRa Packet Structure

A. Basic Observations

For multiple collided packets, we first find the start of each packet. Then, we apply a moving window (the smallest decoding unit containing the collided LoRa signal) aligned with each packet to the signal. As shown in Figure 3, we consider three consecutive windows, each containing 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.

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. The segment of \( R(t) \) contained in window \( w_1 \) can be written as

\[
\begin{align*}
  r_1(t) &= R(t - \tau_1) \\
  &= Ae^{j2\pi f_s (t - \tau_1)} C(t - \tau_1), t \in [\tau_1, T_{chirp})
\end{align*}
\]

(B) Limitations of State-of-the-arts

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). 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

\[
C(t)) \text{ This de-chirp operation can concentrate the signal power in the time domain to a single frequency, and we obtain}
\]

\[
C(t, f_s)C^{-1}(t) = e^{j2\pi f_s t}
\]

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 6~65535 baseline up-chirps to determine the start of the LoRa packet and 2 up-chirps with modulated data to carry extra information. The SFD contains 2.25 baseline down-chirps. The payload contains data bits modulated with up-chirps.

III. Motivation

A. Basic Observations

For multiple collided packets, we first find the start of each packet. Then, we apply a moving window (the smallest decoding unit containing the collided LoRa signal) aligned with each packet to the signal. 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.

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. The segment of \( R(t) \) contained in window \( w_1 \) can be written as

\[
\begin{align*}
  r_1(t) &= R(t - \tau_1) \\
  &= Ae^{j2\pi f_s (t - \tau_1)} C(t - \tau_1), t \in [\tau_1, T_{chirp})
\end{align*}
\]
length of the chirp segment in a window. A long chirp segment can lead to a high peak. 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 cannot be decoded. Our approach leverages the entire chirp in the aligned window to retain the peak height to the greatest extent to avoid the SNR loss.

C. Summary

- 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 if the peak is highest compared with the peaks in the preceding window and following 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 signal, we detect the number of collided packets and the start position of each packet. (2) Collision partition: Then, we apply a moving window to align with each packet and extract peaks after de-chirp and FFT. (3) Aligned chirp detection: By comparing the peak information with adjacent windows, we find the peak of the chirp aligned with each window. (4) Packet separation: We group peaks of aligned chirps based on the position of each packet. (5) CFO elimination: We eliminate the impact of CFO to decode peaks of each packet.

B. Challenges

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

C. Peak Extraction

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. 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 \( \text{mean}(H) \) and \( \text{std}(H) \), where \( \text{std}(H) \) is the standard deviation of \( H \). In traditional outlier detection algorithm, data points that exceed \( \text{mean}(H) + k \cdot \text{std}(H) \) (\( k = 3 \)) 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 \( r = \text{mean}(H) + 6 \cdot \text{std}(H) \) 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 \( H \).

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

D. Packet Start Detection

We leverage the structure of the LoRa packet to detect the accurate start position of each packet in a collision. 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, assume the time offset between the start of the moving window and the start of the packet is $\tau$. 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 $c_1$ 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, for chirp $c_2$ and other baseline up-chirps in a LoRa preamble, the frequency of each peak should be $f = f_s - k\tau$.

By finding $N_p$ consecutive peaks at the same frequency $f$, we can find the preamble and calculate the start of the packet. When there are multiple collided packets, we can find multiple groups of peaks, each group corresponding to a packet. Then we can calculate the start of each packet.

In practice, the existence of CFO introduces errors in packet start detection. Thus, we need to estimate the CFO under 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 (blue line) and with CFO (yellow line). For the baseline up-chirp in the first window with time offset $\tau$, it results in a peak at position $f_1 = f_{sfo} - f_0 + \tau$. For the baseline down-chirp in the second window, it results in a peak at position $f_2 = f_{sfo} + f_0 + \tau$. We can see the time offset leads to the opposite shift to the peak frequency shift centered at CFO. Therefore, CFO can be calculated as $f_{sfo} = f_2 - f_1$ and the time offset can be calculated as $\tau = -\frac{f_2 - f_1}{2k}$.

Then, we consider the case with collision. The key challenge here is that there are multiple peaks in each window, and we need to find the preamble and SFD to the same packet to calculate CFO. When multiple packets collide, we first find the preamble for each packet by finding $N_p$ peaks at the same frequency. Meanwhile, as shown in Figure 8, there are multiple peaks of SFDs as SFDs of different packets are mixed together, leading to difficulty in calculating CFO. Given a peak $P$ of a preamble $PRE$, we find the peak of SFD corresponding to the same packet $pkt$ as follows. For a peak $P_{SFD}$ of $SFD_{i}$ in the mixed peaks of SFDs, we calculate the CFO and $\tau$ by combining $P_{SFD}$ with $P$. We then check whether $P_{SFD}$ and $P$ are from the same packet based on the CFO and $\tau$ as shown in Figure 9. We shift the moving
window by a time offset $\tau$ to align with the packet $pkt$. If $SFD_i$ is from $pkt$, $SFD_i$ should be accurately aligned with the window, and the height of its peak $H_{align}$ should reach its highest height; otherwise, $SFD_i$ should not be aligned with the window. Thus, we slightly shift the moving window by $\delta$ to the left and right, and then calculate the height of peaks corresponding to $SFD_i$. Denote the peak height in the left window and right window as $H_1$ and $H_r$, respectively. $SFD_i$ is from packet $pkt$ iff $H_1 < H_{align}$ and $H_r < H_{align}$. Otherwise, $SFD_i$ is not from the packet $pkt$, and we should test other SFDs. Note that the value of $\delta$ should be smaller than $\text{abs}(f_{closest} - f_{SFD_i})/k$ to guarantee that only $SFD_i$ can be aligned with the window, where $f_{closest}$ is the frequency of the closest SFD peak to $SFD_i$, $f_{SFD_i}$ is the frequency of $P_{SFD_i}$, and $k$ is the frequency changing rate of a chirp.

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 calculate all peaks in those three windows. As shown in Figure 10, 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 $H_i \geq H_{i+1}$.

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. Note when there are multiple consecutive identical chirps in a collided packet and the position of the moving window is after the start of these chirps, there will be two peaks satisfying the frequency and height constraint. 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.

Ideally, the above process assumes that peaks of segments in $w_{i-1}$, $w_i$, $w_{i+1}$ 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 $c_i^l$ ($1 \leq l \leq N-1$) are before $c_i$ and $N-1$ chirps $c_i^r$ ($1 \leq l \leq N-1$) are after $c_i$. Assume $p_i^l$ and $p_i^r$ ($1 \leq l \leq N-1$) are peaks of $c_i^l$ in $w_{i-1}$ and $w_i$, and $H_i^l$ and $H_i^r$ are their height. As the length of the chirp segment of $c_i^l$ in $w_{i-1}$ must be longer than that in $w_i$, we have $H_i^l > H_i^r$. If $p_i^l$ can be extracted in $w_{i-1}$, $p_i^l$ can be extracted in $w_{i-1}$. However, $H_i^l$ and $H_i^r$ do not satisfy the height constraint. Thus, $p_i^l$ are not the peak corresponding to the aligned chirp $c_i$ and can be removed. Similarly, We can remove peaks of $c_i^r$. After removing peaks of those unaligned chirps, the remaining peak is one corresponding to aligned chirp $c_i$.

Example. Figure 10 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 $w_2$. $p_1$ and $p_2$ are peaks of $c_2$ in $w_1$, $w_2$ and $w_3$. Suppose $p_1$ and $p_3$ cannot 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 $c_2$, $p_1$ and $p_3$ correspond to $c_1$ from packet.
Fig. 10. Aligned chirp detection when peaks of chirp segments cannot be detected: we can remove the peaks of unaligned chirps to derive the right one.

1 and $c_2^3$ from packet 3, $p_1^1$ and $p_2^1$ correspond to $c_1^3$ from packet 1 and $c_3^3$ from packet 3. $c_1^3$ and $c_2^3$ are before $c_2^1$ while $c_1^1$ and $c_2^1$ 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 $H_2^b$ denote their height. The segment of $c_1^b$ in $w_1$ is longer than that in $w_2$, i.e., $H_1^b > H_2^b$. $p_1^b$ can be extracted in $w_2$. Thus, $p_1^b$ can be extracted in $w_1$. However, $H_2^b$ 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, $p_2^b$, $p_1^b$, and $p_2^a$ can be removed.

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.

The above 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. For example, we can leverage the small frequency distortion due to hardware imperfection [16]. Thus, we can divide those peaks according to the fractional part of the frequency distortion and then separate into different packets.

F. Packet Separation and CFO Elimination

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. 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. Then we can decode chirps for each packet. Then, we chirps belong to the same packet, we first compensate the CFO and then decode the packet. Assume the demodulated frequency for a chirp $f_s^*$, then the real modulated data after remove CFO is

$$f_s = (f_s^* - f_{cfo}) \mod 2^{SF}$$  \hspace{1cm} (7)

V. 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 GNURadio. The maximum sampling rate is 20 Msamples per second (Mps), and We only use 1 Msps due to the limited bandwidth. In order to better control the LoRa nodes, we implement the function of LoRa nodes on HackRF one. Therefore, we can use GNURadio+HackRF One as a LoRa transceiver to create packet collisions.

**Software:** The HackRF One can provide PHY layer samples of the received signal. We implement AlignTrack in MATLAB on a PC to process LoRa PHY 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.

Note that AlignTrack does not depend on the specific hardware platform. It can decode collision 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 use a HackRF One as the receiver and multiple HackRF One nodes as transmitters. 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 packet collisions.
packets, (2) different SNR, (3) different time offset, and (4) different spreading factors among packets. Further, we compare AlignTrack with the following state-of-the-arts.

- **mLoRa** [12], a LoRa collision decoding approach based on SIC in the time domain.
- **FTrack** [13], time-domain frequency continuity based LoRa collision decoding.
- **CoLoRa** [14], a LoRa collision decoding approach based on the height difference of different packets in the frequency domain.
- **NScale** [15], a LoRa collision decoding approach leveraging a non-linear scaling function in the frequency domain.
- **Choir** [16], a LoRa collision decoding approach based on frequency shift due to imperfection hardware.

Packets sent by each transmitter are known in advance to calculate the SER, BER, and throughput. In order to show the 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) **Impact of the number of packets:** We evaluate the performance of AlignTrack at the different number of overlapping packets. 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 12 shows the averaged SER and BER with the different number of overlapping packets. 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.

2) **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 13 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 6.79%, and the BER is 3.64%, 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 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. This also validates that AlignTrack introduces very small SNR loss.

Figure 14 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 14(a) shows that the SER of AlignTrack is much lower than that of the other methods. When $SNR = 0$ 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 14(b) shows the network throughput. When $SNR \approx -20$ dB, the network throughput of AlignTrack is 57 sps, which is $1.68 \times$ of NScale (34sps), $3.35 \times$ of Choir (17sps) and $3 \times$ of CoLoRa (19sps).

3) **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
to a very small height change of a peak. Thus, we evaluate how
symbol time offset can impact the performance of AlignTrack.

We create different symbol time offsets of a 2-packet
collision. Figure 15 shows the averaged BER under the impact
of symbol time offset and SNR. The BER decreases 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
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.

4) 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.
Figure 16 shows the normalized time consumption under dif-
ferent 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× and 15× 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.

5) Performance in an outdoor network: We evaluate the
performance of AlignTrack in a real outdoor LoRa network.
We use COTS LoRa nodes with chip SX1278 as transmitters
and a HackRF One as the receiver. As shown in Figure 17,
we deploy LoRa nodes in 12 different places. Figure 18 shows
the averaged SER and throughput of four methods when the
number of overlapping packets varies from 1 to 12.

Figure 18(a) shows the averaged SER. The SER increases
with the increasing number of overlapping packets. 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 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. When the number of
overlapping 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 18(a) shows that when there is
no collision, there are still decoding errors in Choir. This is
because Choir separates packets by grouping peaks with the
same fractional part of the frequency of peaks. The fractional
part is mainly determined by the CFO due to the hardware
imperfection. However, the CFO changes with time, and the
fractional part changes with time. And thus, peaks for the same
packet will be divided into different packets. AlignTrack uses
the integer part of the frequency of peaks, and thus can be
more robust to distinguish packets.

Figure 18(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).
VI. 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, which needs a high SNR (e.g., $SNR > 0$ dB). Choir [16] exploits the frequency shift of imperfect hardware to separate packets. CoLoRa [14] considers the height relationship of peaks in adjacent windows. NScale [15] multiplies the chirp segment with a non-stationary scaled down-chirp and leverages peaks of different segments to distinguish different packets. CoLoRa and NScale need to partition a chirp and introduce inevitable SNR loss in practice. AlignTrack leverages the entire chirp in LoRa and introduces a very small SNR loss. The SNR loss introduced by CoLoRa and NScale requires LoRa nodes to transmit LoRa packets with higher strength, while AlignTrack 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) [17], [18]. 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 [19] [20] and RTS-CTS [21] [22]. 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. Channel Activity Detection (CAD) method introduced by LoRa needs to detect the LoRa preamble, which introduces a high overhead and power consumption. Meanwhile, CSMA/CA significantly reduces the transmission efficiency especially considering the dense deployment of LoRa nodes and the long communication distance. The impact of the hidden/exposed terminal problems is exacerbated and will significantly reduce the network efficiency. SIC [23] is also a general method for collision resolution. SIC 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 LPWANs. A variety of works have been proposed to improve the performance of LPWANs. OPR [24] uses multiple gateways to recover packets subject to CRC and/or FEC errors. Chime [25] analyzes the path signals traverse from the client to distributed and coordinated gateways to choose the optimal frequency of the LPWAN client. LiteNap [26] enables sub-Nyquist sampling and packet decoding to improve energy efficiency. EF-LoRa [27] allocates different network resources to achieve fair energy consumption among end devices. The work [28] conducts a series of experiments to verify the claims made by Semtech on LoRa.

VII. 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 $1.68 \times$ compared with NScale and $3 \times$ compared with CoLoRa.

ACKNOWLEDGMENT

We thank our shepherd I-Hong Hou and anonymous reviewers for their insightful comments to improve the quality of our work. This work is in part supported by NSFC No. 61932013 and Tsinghua-Foshan Innovation Special Fund (TFISF). Jiliang Wang is the corresponding author.

REFERENCES


