FTrack: Parallel Decoding for LoRa Transmissions

Xianjin Xia, *Member, IEEE*, *ACM*, Yuanqing Zheng, *Member, IEEE*, *ACM*, and Tao Gu, *Senior Member, IEEE, Member, ACM*

***Abstract*— LoRa has emerged as a promising Low-Power Wide Area Network (LP-WAN) technology to connect a huge number of Internet-of-Things (IoT) devices. The dense deployment and an increasing number of IoT devices lead to intense collisions due to uncoordinated transmissions. However, the current MAC/PHY design of LoRaWAN fails to recover collisions, resulting in degraded performance as the system scales. This article presents FTrack, a novel communication paradigm that enables demodulation of collided LoRa transmissions. FTrack resolves LoRa collisions at the physical layer and thereby supports parallel decoding for LoRa transmissions. We propose a novel technique to separate collided transmissions by jointly considering both the time domain and the frequency domain features. The proposed technique is motivated from two key observations: (1) the symbol edges of the same frame exhibit periodic patterns, while the symbol edges of different frames are usually misaligned in time; (2) the frequency of LoRa signal increases continuously in between the edges of symbol, yet exhibits sudden changes at the symbol edges. We detect the continuity of signal frequency to remove interference and further exploit the time-domain information of symbol edges to recover symbols of all collided frames. We substantially optimize computation-intensive tasks and meet the real-time requirements of parallel LoRa decoding. We implement FTrack on a low-cost software defined radio. Our testbed evaluations show that FTrack demodulates collided LoRa frames with low symbol error rates in diverse SNR conditions. It increases the throughput of LoRaWAN in real usage scenarios by up to 3 times.**

***Index Terms*— Internet of Things, LoRaWAN, collision resolving, parallel decoding.**

I. Introduction

RECENTLY , LoRa has emerged as a promising technology for Low-Power Wide Area Networks (LP-WANs). Among many LPWAN technologies (e.g., SigFox [1], NB-IoT [2]), LoRa technology [3] has attracted wide attention due to its low cost, long communication range, and supports from IoT industry as well as open-source and research community. LoRaWAN is an open-standard networking layer governed by the LoRa Alliance [3], which has about 400 member companies including Tencent, IBM, Cisco, Semtech, *etc*. LoRa employs a variant of Chirp Spread Spectrum (CSS) modulation to support several kilometers of wireless transmissions at very low power consumption. The CSS modulation of LoRa is robust against interference, noise, multi-path and Doppler effects. Such characteristics make LoRaWAN a promising communication technology for IoT innovations such as smart city, health care, environment monitoring, warehouse management, *etc*. When a LoRa node receives a packet, it first detects the preamble, then searches for chirp boundaries (*i.e*., symbol edges) and locates the chirp signal of each symbol to decode. The standard demodulation scheme dechirps the incoming signals by multiplying with a down chirp, then performs FFT on the multiplication, which produces a single FFT peak indicating the symbol. The PHY technique of LoRa (*i.e*., CSS) is inherently robust to interference and noise in the ISM band (*e.g*., WiFi, RFID, *etc*.).

Although LoRa uses several PHY techniques for parallel transmissions of multiple LoRa nodes, a high-end LoRa gateway can only support up to 8 LoRa nodes to transmit at the same time [4]. It fails to meet the need of many IoT applications which require dense deployment of LoRa devices in practice. Constrained by the hardware capability and power supply, LoRa nodes typically adopt a simple aloha-based MAC for collision avoidance. As a result, collisions may occur among LoRa nodes with the same PHY configuration. One may configure LoRa nodes with different radio parameters (*e.g*., channel, spreading factor, *etc*.) to mitigate collision, but it requires cooperation among different operators and service providers. A more effective method is to enable parallel decoding for LoRa transmissions without any extension to COTS LoRa nodes or any coordination among the users. Ideally, a LoRa node should be able to join on-going communications in parallel with other nodes without specific coordination.

To enable parallel decoding for LoRa transmissions, prior work (*e.g*., Choir [5]) exploits the frequency offset introduced by radio hardware to separate collided frames. The key idea is to classify the collided frames according to the distinct frequency offset of each LoRa node (*e.g*., carrier frequency). However, due to background noise and phase jitters, it is difficult to accurately extract the hardware-induced offset of carrier frequencies in practice. Moreover, the carrier frequencies of LoRa nodes inevitably resemble each other as the number of LoRa nodes increases. Consequently, Choir may incorrectly classify collided frames and result in decoding errors.

In this article, we present FTrack, a parallel decoding scheme for LoRa that resolves collision by jointly considering both time and frequency domain features. We exploit an observation that, as the airtime of a low-rate LoRa packet is

Manuscript received October 24, 2019; revised July 19, 2020; accepted July 22, 2020; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor B. Shrader. Date of publication August 27, 2020; date of current version December 16, 2020. This work was supported in part by the National Nature Science Foundation of China under Grant 61702437; in part by the Hong Kong GRF under Grant PolyU 152165/19E; and in part by the Australian Research Council (ARC) Discovery Project under Grant DP190101888 and Grant DP180103932. *(Corresponding author: Yuanqing Zheng.)*

Xianjin Xia and Yuanqing Zheng were with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. They are now with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: csxxia@comp.polyu.edu.hk; csyqzheng@comp.polyu.edu.hk).

Tao Gu was with the School of Computer and IT, RMIT University, Melbourne, VIC 3000, Australia. He is now with the School of Computer Science and Software Engineering, RMIT University, Melbourne, VIC 3000, Australia (e-mail: tao.gu@rmit.edu.au).

Digital Object Identifier 10.1109/TNET.2020.3018020

much longer than the packets in other wireless technologies (*e.g.*, WiFi and RFID), LoRa packets are more likely to be misaligned and separated in the time domain. Due to the periodicity of LoRa symbols, the symbols from the same frame share the same pattern of edge misalignment in time. We leverage such time-domain information to separate collided LoRa symbols. In addition, we also leverage the fact that the chirp of a LoRa symbol starts from an initial frequency and then increases continuously along a fixed 'track' to sweep through the whole LoRa bandwidth. The different arrival time of colliding frames would result in timing offset and a corresponding shift in frequency between collided symbols. Thus, chirps of collided symbols are also likely to exhibit as different tracks in the frequency domain. We separate the frequency tracks of individual chirps to recover collisions of all frames.

Intuitively, the decoding process consists of the following key steps. FTrack first detects the edges of LoRa symbols in the time domain and groups the symbols according to different LoRa nodes. Then, it iteratively demodulates and decodes the symbols of each individual LoRa node. FTrack is orthogonal to existing parallel decoding techniques for LoRa transmissions. As long as the transmissions are interleaved with misaligned symbol edges, LoRa nodes should be able to transmit concurrently even with the same parameter configuration in the same channel.

Turning the idea into reality, however, entails tremendous challenges. First, it is non-trivial to detect the symbol edges of each individual transmission from collided signals and group the symbol edges according to each LoRa node. To address this problem, we leverage the fact that the preamble of a LoRa node is known in advance, which allows us to extract the symbol edges from collided signals by correlating the known preamble with the collided signals. The correlation peaks indicate the symbol edges in the preamble. The extracted symbol edges, however, are interleaved in the time domain due to collisions. As a LoRa preamble consists of repeated base chirps, the signal frequency would increase continuously across boundaries of all chirps associated with the preamble, resulting in a long track of continuous frequency. As such, we filter out any interfering chirps not belonging to the long frequency track to extract a pure collision-free preamble. We detect symbol edges from the extracted preamble and use that timing information to accurately pinpoint the symbol edges in its payload. Between the symbol edges corresponding to the same LoRa node, the frequency of a LoRa chirp should continuously increase, while the frequencies of other coexisting chirps may exhibit sudden changes. We leverage such a fact to filter out coexisting chirps and resolve collisions. We iterate the above process until all chirps are correctly associated with their corresponding nodes.

Another practical challenge arises from the requirements of decoding concurrent transmissions in real-time. Our measurement results show that the computation overhead of FTrack is dominated by frequency track extraction which requires to perform FFT on per-sample basis with a sliding window. To avoid the costly FFTs, we leverage the overlapping of PHY samples between successive windows to compute the FFT of a new window by updating the FFT of the previous window. Besides, we optimize edge detection by avoiding the computation-intensive correlation operations. We use the specific frequency-time relationship of chirps in LoRa preamble to extract the timing of symbol edges directly from the frequency track of preamble.

We implement FTrack using software defined radios (SDRs). To reduce the deployment cost, we employ low-cost SDRs to collect PHY samples and run the proposed parallel decoding scheme. We build a testbed of 20 LoRa nodes to evaluate FTrack with a variety of transmission configurations. Results show that FTrack decodes collided frames with low symbol error rates in diverse SNRs. Compared with the state-of-the-art schemes, FTrack improves the throughput of LoRaWAN networks in real usage scenarios by up to 3 times.

II. Background and Motivation

LoRa adopts Chirp Spread Spectrum (CSS) as the PHY modulation scheme. Symbols are modulated as up-chirp signals whose frequencies increase linearly with time over a predefined bandwidth. LoRa varies the initial frequency of an up-chirp to modulate different data. Such a procedure can be represented as follows.

$$S(t, f_{sym}) = e^{j2\pi(\frac{k}{2}t+f_0)t} \cdot e^{j2\pi f_{sym}t} = C(t) \cdot e^{j2\pi f_{sym}t}, \tag{1}$$

where f_{sym} denotes the initial frequency of the up-chirp (*i.e.*, encoded symbol). $C(t) = e^{j2\pi(\frac{k}{2}t+f_0)t}$ represents the raw chirp signal (termed *base chirp*); f_0 and k denote the starting frequency and frequency increasing rate of the chirp, respectively.

A LoRa receiver can demodulate an incoming chirp as follows. It first multiplies the received signal with the *conjugate of the base chirp* denoted as $C^{-1}(t)$, performs a Fast Fourier Transform (FFT) on the multiplication, and searches for power peak in the FFT bins to demodulate symbol. The procedure can be represented as follows

$$S(t, f_{sym}) \cdot C^{-1}(t) = e^{j2\pi f_{sym}t} \tag{2}$$

The FFT of $e^{j2\pi f_{sym}t}$ produces one peak in the FFT bins, that indicates the frequency component of f_{sym}.

When multiple LoRa nodes transmit concurrently, their signals add up at the receiver. Figure 1 presents the signal received with USRP N210 when two LoRa nodes transmit simultaneously. In the figure, we observe multiple chirps overlapping in time, each of which corresponds to the symbol of one transmitter. As a standard LoRa demodulator (Eq.(2)) searches for the maximum in FFT results, it cannot reliably decode the collided signals. We aim to support parallel transmissions of multiple LoRa nodes and resolve collisions.

A recent work, Choir [5], exploits the frequency offset (*i.e.*, offset of carrier frequency) introduced by the hardware imperfection and diversities of LoRa nodes to separate collided frames. However, it is difficult to extract tiny hardware frequency offset in the presence of noises and inter-packet collisions. More importantly, as the number of LoRa nodes increases, the carrier frequencies of nodes are likely to resemble each other. Solely relying on hardware-induced frequency offsets may fail to differentiate LoRa nodes in practice.

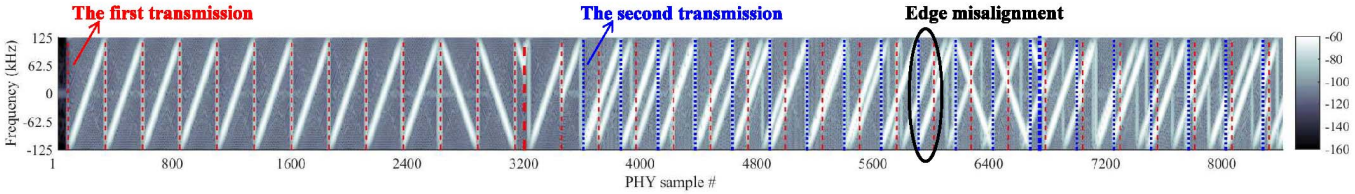
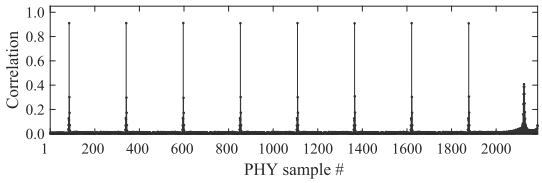
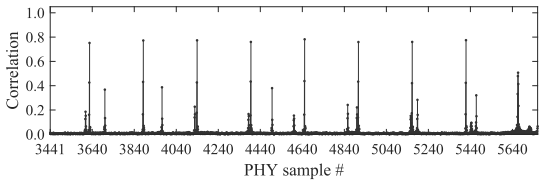


Fig. 1. Collision of two LoRa transmissions. Red/blue dashed lines indicate the symbol edges of two LoRa transmissions.



(a) Correlation detection on preamble of the first transmission



(b) Correlation detection on preamble of the second transmission

Fig. 2. Edge detection on (a) collision-free and (b) colliding preambles, *i.e.*, the preambles of the first and second transmissions in Fig.1. The correlation peaks indicate the boundaries of base chirps, *i.e.*, symbol edges.

In this article, we leverage the fact that the concurrent transmissions of LoRa packets are likely to be misaligned in time. Figure 1 illustrates the time misalignment of two collided LoRa transmissions. A receiver can increase the sampling rate to better differentiate the misaligned chirps in the time domain. Besides, as the frequencies of LoRa chirps vary linearly with time, the time misalignment of chirps leads to the corresponding frequency offsets between the chirps. Thus, we can separate collided chirps in both time and frequency domains. We configure COTS LoRa nodes to transmit with the default parameters (*i.e.*, Spreading Factor=8, bandwidth=250kHz) and use a low-cost receive-only SDR (*e.g.*, RTL-SDR dongle) to sample at a high rate (*e.g.*, 1MS/s). The receiver chain will decimate the received LoRa packets and produce 256 PHY samples for each symbol. As the data rate of LoRa node is low, the high sampling rates of SDRs provide sufficient time resolution to separate misaligned chirps.

III. Design

A. Detecting Symbol Edges

A LoRa receiver must accurately locate the boundaries of a chirp (*i.e.*, symbol edges) to demodulate the chirp. For example, Fig. 3(a) illustrates three base chirps that start with the same initial frequency f_0. If the chirp boundary t_0 is correctly located, the initial frequency can be correctly measured as f_0 and the base chirp can be demodulated. However, if the boundaries are located mistakenly with an offset of Δt, where $t_1 = t_0 + \Delta t$, the measured initial frequency would become $f_1 = f_0 + \Delta f$, leading to demodulation errors. The increasing rate of chirp frequency $\frac{k}{2} = \Delta f / \Delta t$ depends on the spreading factor of LoRa modulation which remains constant during the transmission of a packet.

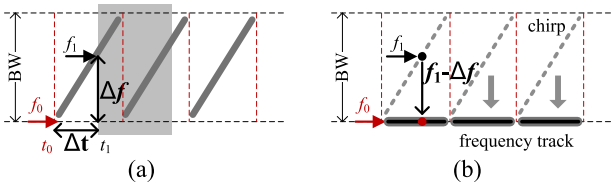
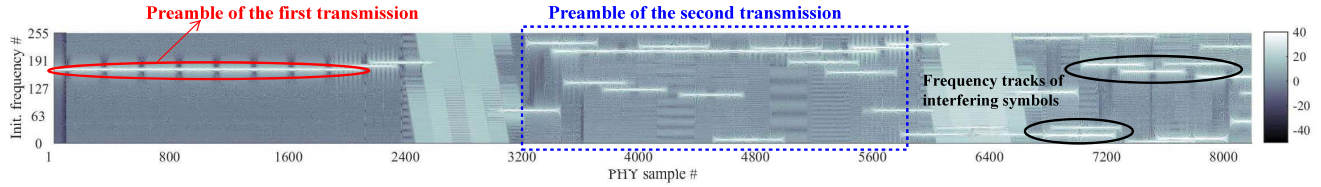


Fig. 3. Illustration of (a) preamble chirps, and (b) the corresponding frequency tracks, *i.e.*, normalize instantaneous frequency into the meta initial frequency.

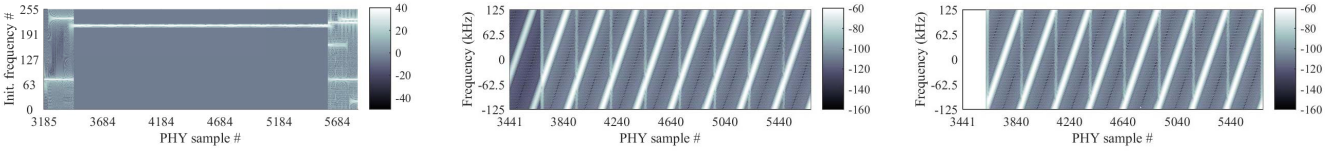
Intuitively, we detect chirp boundaries and calibrate the frequency offset introduced by timing misalignment by leveraging our prior knowledge of LoRa preamble which is a standard method in LoRa demodulation. We calculate the correlation of received signals with a base chirp to detect symbol edges. When we apply the method on a non-collided preamble (*e.g.*, the first preamble in Fig. 1), symbol edges can be well identified by the correlation peaks as shown in Fig. 2(a). However, when collision happens (*e.g.*, the second preamble in Fig. 1), the concurrent LoRa transmissions may introduce interference as shown in Fig. 2(b). As such, a key problem arises: *how shall we detect symbol edges in the presence of collisions?*

1) Extracting Interference-Free Preambles: To solve the problem of inter-packet interference, we extract pure preamble from collisions and then detect symbol edges with the extracted interference-free preamble. The method is inspired by the following observation. The frequency of a LoRa chirp will start from an initial value (named *meta initial frequency*) and increase linearly with time. All chirps in the preamble have the same meta initial frequency (*i.e.*, f_0 in Fig. 3(a)), yet those from the payload of LoRa frame do not. We exploit this fact to detect LoRa preamble.

Ideally, the meta initial frequency of a LoRa chirp (*i.e.*, f_0) should be measured at the arrival time of the chirp (*i.e.*, chirp boundary t_0). Without knowing the exact boundary of the chirp, we may obtain different initial frequency (*e.g.*, f_1) when measuring at different locations (*e.g.*, t_1) of the chirp, as illustrated in Fig. 3(a). Indeed, f_1 corresponds to the instantaneous frequency of the chirp at the measuring point, but not the meta initial frequency (*i.e.*, f_0). To deduce f_0, we leverage the relationship between chirp frequency and time to *remove the frequency deviation (i.e., Δf) induced by time offset* Δt, as illustrated in Fig. 3(b). Doing so, we normalize the time-varying frequency of chirps (*e.g.*, f_1) into horizontal frequency lines, termed as *frequency tracks*, that indicate the meta initial frequency of chirps, *i.e.*, f_0. With such a normalization process, we can measure the meta initial frequency at any time of a chirp.



(a) The normalized initial frequencies (*i.e.*, frequency tracks) of collided LoRa chirps



(b) Filtering out the interfering chirps (c) Reconstructing preamble from offset #3441 (d) Reconstructing preamble from offset #3641

Fig. 4. Extraction of collision-free preambles: (a) normalize LoRa chirps into meta initial frequency and transform colliding symbols into different frequency tracks, (b) remove interfering symbols, and (c-d) use inverse operations (*i.e.*, the inverse of frequency normalization, followed by an inverse FFT) to reconstruct the second preamble.

We employ a sliding window on the received signals shown in Fig. 1 to extract the normalized initial frequency of chirps. As shown in Fig. 4(a), the normalized initial frequency of each LoRa chirp corresponds to a horizontal high-power line (*i.e.*, frequency track). The long frequency tracks correspond to LoRa preamble. Since the base chirps in preamble have identical meta initial frequency, their frequency tracks form a single long track as illustrated in Fig. 4(a). We exploit such a property to detect LoRa preambles.

From Fig. 4(a), we also observe that the preambles of collided frames are separated into different frequency tracks. The location of frequency track is determined by both the meta initial frequency and the arrival time of a LoRa chirp (see § III-C for detail). Therefore, although the chirps of two preambles have the same meta initial frequency, collided preambles will be separated into different tracks as long as their chirp boundaries are interleaved in time. We exploit the property to extract LoRa preamble from collisions. Specifically, we filter out interference by setting the frequency tracks of interfering chirps to zero while only keeping the long frequency track as shown in Fig. 4(b). As the extracted frequency track indicates the normalized initial frequency of base chirps in preamble, we employ an inverse operation to reconstruct a pure, interference-free preamble, as shown in Fig. 4(c). We can reconstruct the same preamble using a segment of the frequency track starting from different offsets, as shown in Fig. 4(c) and (d). We then perform correlation detection on the reconstructed preamble to detect symbol edges. In this way, we eliminate the impact of collision and detect symbol edges of concurrent transmissions.

Fig. 5 summarizes the key steps of edge detecting. Firstly, we measure the instantaneous frequency of chirps at any time offset in Step 1. Step 2 normalizes the measured frequency to chirps' meta initial frequency (*i.e.*, frequency tracks). In Step 3, we check the length of frequency tracks to detect preamble. We identify the frequency bin and arrival time of each preamble. Step 4 uses the information to filter the frequency tracks of interfering chirps. We extract interference-free preambles in the frequency domain and convert to time domain for edge detecting in Steps 5, and detect edges using correlation in

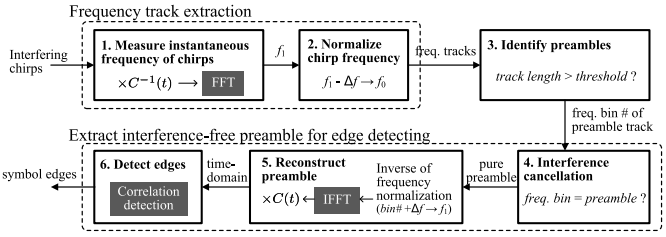


Fig. 5. The work flow of edge detection: symbol edges are detected from the interference-free preamble.

Step 6. The computational overhead mainly stems from FFT operation and correlation detection. The time complexity is $O(n^2)$, where n denotes the number of samples in a chirp.

2) Optimizing Edge Detection: In the edge detection process (Step 4–6), the FFT operation and the correlation detection dominate the computational overhead. In order to support real-time parallel decoding of LoRa transmissions, we propose an optimization to avoid the costly the FFT operation and the correlation detection by leveraging our prior knowledge of chirps in LoRa preamble.

Our optimized method of edge detection leverages the fact that the instantaneous frequency of a chirp increases linearly with time. Once the frequency track of a preamble has been identified (*i.e.*, *freq. bin #* in Step 3 of Fig. 5), we can employ an inverse operation of frequency normalization to extract the instantaneous frequency (say f_1) of chirp at a particular time offset t_1. Then we determine symbol edge based on the frequency-time relationship of LoRa chirp $f_1 - f_0 = \frac{k}{2}(t_1 - t_0)$, as illustrated in Fig. 3(a), where t_0 is the edge and f_0 the initial frequency of preamble chirp. As f_1 and t_1 have been obtained, while f_0 and k (*i.e.*, frequency increasing rate) are known based on our prior knowledge of preamble, we can directly extract t_0, *i.e.*, the timing of symbol edge, from the frequency track of preamble. This optimized edge detection method avoids the computation-intensive correlation operation.

B. Demodulating Symbols

To demodulate symbols in the payload of a LoRa frame, we need to first locate the chirp of each symbol. We leverage

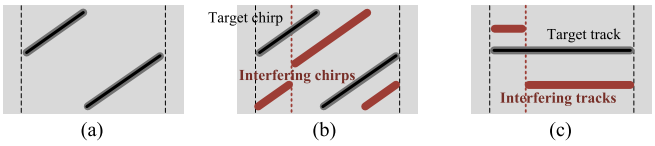
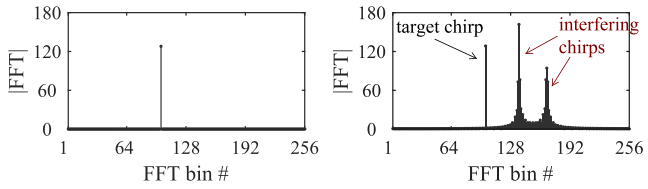


Fig. 6. Symbol demodulation: (a) locate a target chirp within a demodulation window, (b) a target chirp and an interfering chirp within the same demodulation window due to collisions, (c) filter out the interfering chirp by detecting frequency continuity.



(a) Demodulation without collisions (b) Demodulation with collisions

Fig. 7. Demodulation with interference: the target chirp can be demodulated incorrectly due to interference.

the timing information of symbol edges detected from a LoRa preamble to deduce symbol edges in the payload. We refer to the signals located between two symbol edges as a *demodulation window*. Ideally, when there is no collision, there will be only one complete chirp (*i.e.*, *target symbol*) in a demodulation window, as illustrated in Fig. 6(a). We can demodulate the chirp using the standard procedure presented in § II, which will produce a single FFT peak as illustrated in Fig. 7(a). However, when collision happens, chirps of interfering transmissions would also appear in the demodulation window, as shown in Fig. 6(b). Directly applying the standard algorithm to demodulate the collided chirps will produce multiple FFT peaks, as shown in Fig. 7(b), where the highest peak may not correspond to the target symbol. Therefore, how to correctly demodulate the target symbol in the presence of collisions becomes a key issue.

Our solution separates the target symbol from collisions by jointly exploiting the frequency-domain and time-domain features of concurrent LoRa transmissions. We find that the frequency of the target chirp changes continuously in the demodulation window, as there is only one chirp of the target symbol in the window. Whereas for interfering symbols, as their symbol edges are misaligned with the demodulation window of the target symbol, the window spans across two chirps. Since the second chirp may change its initial frequency to modulate a different symbol, it will result in a sudden change of chirp frequency. Based on this observation, *we examine the continuity of frequency tracks to filter out interfering symbols in each demodulation window.* As illustrated in Fig. 6(c), the normalized initial frequency of interfering symbols will switch to a different track. In contrast, the target symbol produces a single complete frequency track that spans over the whole window. A complete frequency track within the demodulation window indicates the initial frequency (*i.e.*, encoded data) of the target symbol.

In case that two successive chirps of interference packet carry the same data, their frequency tracks also exhibit continuity in the demodulation window (*e.g.*, Win #1 in Fig. 8). In this case, we need to observe with a longer window (*e.g.*, over two windows). Although the frequency tracks of interference chirps are continuous in the first demodulation window, we can observe discontinuity in the second window, as illustrated in Fig. 8. In contrast, the frequency tracks of target symbols are continuous within both demodulation windows. We can filter out interference by checking the discontinuity point of frequency track against the target's symbol edges.

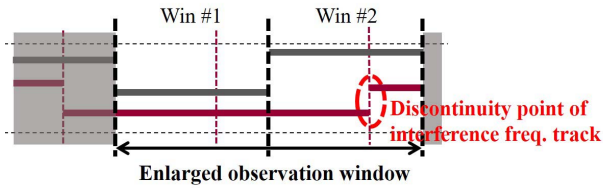


Fig. 8. When two successive interfering chirps carry identical data, frequency discontinuity of interference track (red) is observed in Win #2.

C. Frequency Track in Practice

Frequency track plays important roles in collision resolving: Firstly, it separates colliding symbols into different tracks, indicating different meta initial frequencies. We exploit this property for collision recovery. Secondly, we can directly demodulate symbols from frequency tracks because data are encoded by the meta initial frequency of LoRa chirps (§ II). In this subsection, we describe how to extract frequency tracks from received signals.

1) Considerations of Method Design: The signal of a LoRa chirp is represented by $S(t, f_{sym})$ in Eq. (1). Based on Eq. (1), we can formally represent the frequency track of chirp $S(t, f_{sym})$ as follows

$$F_{track}(f_{sym}) = e^{j2\pi f_{sym} t}, \tag{3}$$

where f_{sym} is the meta initial frequency of the chirp. To extract frequency track from the received signal of a LoRa chirp, one approach is to measure the instantaneous frequency of the chirp and subtract a frequency deviation Δf from the instantaneous frequency, as illustrated in Fig. 3(b). In order to measure the instantaneous frequency, we can use the short-time Fourier transform (STFT) to analyze the changing spectra of signal over time (*i.e.*, spectrogram of signal). However, such a method only provides a coarse-grained estimate of chirp frequency, which cannot precisely identify the frequency track (*i.e.*, meta initial frequency) of LoRa chirp.

As STFT takes PHY samples from received signal with a *PHY Win* to analyze the frequency components, the size of *PHY Win* affects the resolution of frequency estimation. Fig. 9(a-c) present the resulting spectrograms of the same signal with different configurations of *PHY Win*. Comparing Fig. 9(b) and (c), we see that the measured frequency of chirps become narrower with a smaller *PHY Win* of 32, because a smaller *PHY Win* would contain *PHY samples with less time-varying frequencies*. Yet, the chirps do not become thinner as *PHY Win* further decreases from 32 to 8 (see Fig. 9(a) and (b)). The reason is that, when *PHY Win* is too short, STFT must pad a number of zeros to the PHY samples before FFT analysis, which will produce a wide main-lobe

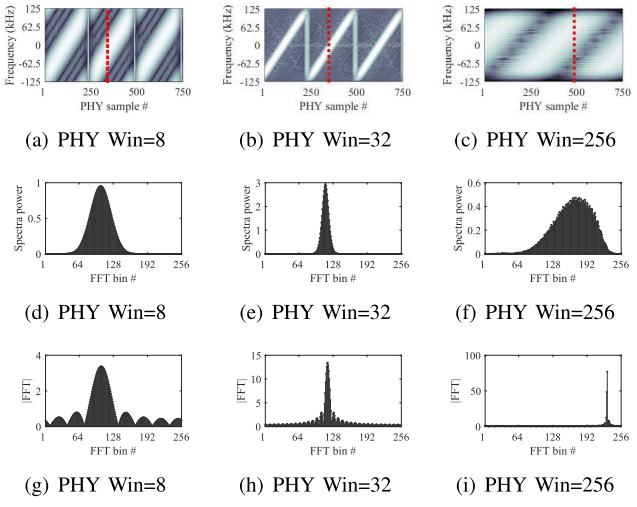


Fig. 9. Frequency estimation with different *PHY Wins* (the *FFT Win* is 256). (a-c): Spectrogram of preamble chirps; (d-f): Width of the FFT peak of *original signal*; (g-i): Width of the FFT peak of *dechirped signal*.

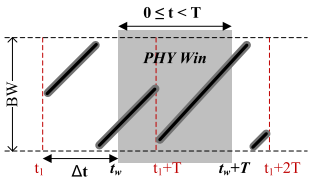


Fig. 10. Illustration of chirp signals in a *PHY Win*.

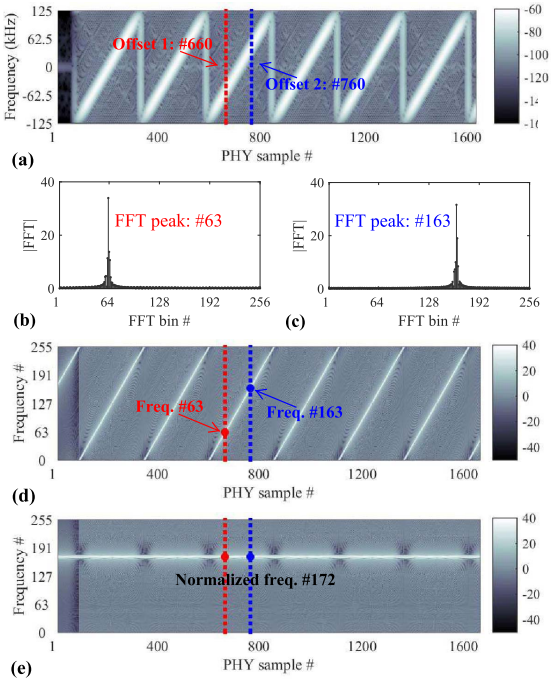


Fig. 11. Steps of extracting frequency tracks: (a) spectrogram of received signals (the preamble part), (b-c) dechirp the signal in *PHY Win* (FFT of dechirped signals at #660 and #760), (d) slide *PHY Win* across all samples (the frequency of dechirped signal vs. offsets of *PHY Win*), (e) remove the frequency deviation induced by t_w (*i.e.*, the time offset of *PHY Win*) to produce the normalized frequency of LoRa chirps, *i.e.*, frequency tracks.

and many sinc side-lobes between two main chirps, as shown in Fig. 9(a). As displayed in Fig. 9(d-f), even with the finest configuration for spectrogram (*i.e.*, *PHY Win*=32), the width of obtained chirp frequency would still span tens of FFT bins, which does not provide sufficient resolution to precisely locate the frequency track of the chirp.

We aim to extract narrow frequency tracks that can precisely indicate the meta initial frequencies of LoRa chirps. As we mentioned, due to the CSS modulation scheme of LoRa, signals in a period of Δt would sweep a frequency width of Δf (see Fig. 3(a)). The time-varying frequencies of chirp within *PHY Win* prevent spectrogram from producing narrow frequency tracks. To solve the problem, we dechirp the signal in *PHY Win* by multiplying with a down chirp (*i.e.*, $C^{-1}(t)$, see Eq. (2)). Fig.s 9(g-i) present the FFT of the signal samples in *PHY Win* after being dechirped. The resulting FFT peaks become narrower and can precisely pinpoint the frequency of corresponding chirps shown in Fig. 9(d-f). Besides, the width of FFT peak becomes thinner as *PHY Win* increases, because the window includes more *PHY samples with the same meta initial frequency*. In particular, when *PHY Win*$=256$, *i.e.*, containing all samples of a chirp, the resulting peak width can reach the resolution of one FFT bin, as shown in Fig. 9(i).

2) Extracting Frequency Tracks: Our approach of frequency track extraction involves three key steps: Firstly, we dechirp the signal samples in *PHY Win* and perform FFT on dechriped signals to measure the instantaneous frequency of chirps at a given time offset. Secondly, we normalize the instantaneous frequency measured at different offsets by subtracting the corresponding frequency deviation. Finally, we move *PHY Win* to the next PHY sample of received signals and repeat the above two steps. We describe in details in the following.

Suppose that n LoRa nodes transmit simultaneously. Let $X(t)$ denote the signal received by a LoRa device and $x_i(t)$ denote the signal from node i, *i.e.*, $X(t) = \sum_{i=1}^{n} x_i(t)$. We focus on the signal of one transmitter (*i.e.*, $x_i(t)$ of node i) in the following. In particular, we configure the length of *PHY Win* to T, *i.e.*, the duration of a LoRa chirp. Let t_w denote the offset position of *PHY Win* in $X(t)$. A *PHY Win* generally spans two LoRa symbols as illustrated in Fig. 10, where the starting edges of the two symbols are represented by t_1 and t_1+T, respectively. We denote the offset between t_1 and *PHY Win* by Δt, *i.e.*, $\Delta t = t_w - t_1$. Specifically, $0 \le \Delta t < T$ (note that when $\Delta t = 0$, only the first symbol is included in *PHY Win*, which is a special case). Let t denote the relative time of chirp signals within *PHY Win*, *i.e.*, $0 \le t < T$. As illustrated in Fig. 10, signals in *PHY Win* are composed of two segments from the first and the second symbols, respectively. Based on Eq. (1), we denote the first segment of signal in *PHY Win* as

$$x_i(t) = h_i \cdot S(t + \Delta t,\ f_1), \quad t \in [0, T - \Delta t) \tag{4}$$

where f_1 represents the initial frequency of the first chirp. h_i is a complex value representing amplitude and phase changes on the wireless channel between transmitter and receiver.

According to the CSS modulation, a time shift of Δt in chirp signals would result in a frequency offset of $\Delta f = \frac{k}{2}\Delta t$ as illustrated in Fig. 3(a). Therefore, Eq. (4) can be transformed as follows

$$x_i(t) = h_i \cdot S(t,\ f_1 + \frac{k}{2}\Delta t), \quad t \in [0, T - \Delta t) \tag{5}$$

Similarly, the second segment of signal in *PHY Win* can be represented by

$$\begin{aligned} & x_i(t) = h_i \cdot S(t-(T-\Delta t),\ f_2) \\ \Leftrightarrow\ & x_i(t) = h_i \cdot S(t,\ f_2 - \frac{k}{2}(T-\Delta t)), t \in [T-\Delta t, T) \end{aligned} \tag{6}$$

where f_2 represents the initial frequency of the second symbol chirp. For conciseness, we focus on the first segment in the following. Similar results are obtained for the second segment.

We dechirp $x_i(t)$ in Eq. (5) by multiplying a standard down chirp (*i.e.*, $C^{-1}(t)$) as follows

$$x_i(t)\cdot C^{-1}(t) = h_i \cdot \underbrace{e^{j2\pi(\frac{k}{2}\Delta t)t}}_{\text{Offset of } \textit{PHY Win}} \cdot \underbrace{e^{j2\pi f_1 t}}_{\text{Initial freq.}},\ t\in[0, T-\Delta t) \tag{7}$$

The FFT of Eq. (7) produces a peak at $f_1 + \frac{k}{2}\Delta t$, which corresponds to the frequency of the first chirp with time shift Δt, *i.e.*, instantaneous frequency of the chirp in *PHY Win*. Fig. 11(d) presents the FFT of dechirped signals when *PHY Win* slides across the PHY samples of chirps shown in Fig. 11(a). It produces thin bright lines that precisely indicate the instantaneous frequency of chirps.

Lastly, we normalize the instantaneous frequency of chirps to extract frequency tracks. We need to subtract $\Delta f = \frac{k}{2}\Delta t$ from the resulting frequency of the dechirped signal in Eq. (7), *i.e.*, $f_1 + \frac{k}{2}\Delta t$. Note that $\Delta t = t_w - t_1$, where t_1 denotes the starting edge of symbol chirp f_1. As t_1 is not known, we cannot directly remove Δt from Eq. (7). Instead, we subtract t_w, *i.e.*, the offset of *PHY Win*, from the second term of Eq. (7) as follows

$$\begin{aligned} F_{norm}(f_1, t_1) &= h_i \cdot e^{j2\pi(-\frac{k}{2}t_1)t} \cdot e^{j2\pi f_1 t} \\ &= h_i \cdot F_{edge}(t_1) \cdot F_{track}(f_1) \end{aligned} \tag{8}$$

where $F_{norm}(f_1, t_1)$ represents the normalized frequency of the first chirp in *PHY Win*, f_1 and t_1 are the meta initial frequency and arrival time of the chirp, respectively. $F_{track}(f_1)$ denotes the ideal frequency track of the chirp, as defined in Eq. (3). $F_{edge}(t_1)$ denotes the frequency shift induced by t_1, *i.e.*, the arrival time (symbol edge) of the first chirp. Since t_1 is invariant as *PHY Win* slides across the signal samples, $F_{edge}(t_1)$ is a constant.

Fig. 11(e) presents the normalized frequency of the dechirped signals shown in Fig. 11(d). Note that, with Eq. (8), we actually normalize the instantaneous frequency of chirp signals into a different frequency track (*i.e.*, $F_{norm}(f_1, t_1)$) that shifts from the ideal frequency track (*i.e.*, $F_{track}(f_1)$) with a specific offset $F_{edge}(t_1)$. Since $F_{edge}(t_1)$, *i.e.*, the frequency offset between $F_{track}(f_1)$ and $F_{norm}(f_1, t_1)$, remains constant across all symbols in the received signal, we can employ $F_{norm}(f_1, t_1)$ to separate collided symbols in practice. Moreover, as $F_{norm}(f_1, t_1)$ is determined by both the meta initial frequency (f_1) and arrival time (t_1) of LoRa chirps, collided symbols would be separated into different frequency tracks as long as their edges (*i.e.*, t_1) are interleaved in time.

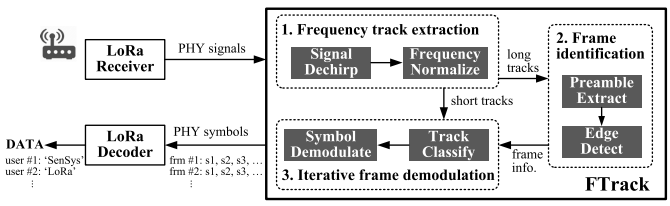


Fig. 12. The workflow of FTrack system. Long track and short track represent the frequency tracks of LoRa preamble and payload symbol, respectively.

3) Optimizing for Real-Time Processing: The basic method of frequency track extraction needs to slide *PHY Win* across all samples and perform FFT on a per-sample basis. Let N_s denote the number of PHY samples and n_{fft} the FFT window size. The computational complexity is $O(N_s n_{fft} \log(n_{fft}))$. Notice that as we slide *PHY Win* on a per-sample basis, the signal samples of a *PHY Win* overlap with that of the former window. We leverage the similarity between the signal samples of two PHY windows to optimize the computation efficiency of sliding FFT.

Without loss of generality, we denote the signal samples of *PHY Win* at time t_w and t_{w+1} by X_w and X_{w+1}, respectively. $X_w = \{x_0, x_1, \cdots, x_{n-1}\}$ and $X_{w+1} = \{x_1, x_2, \cdots, x_n\}$. X_w overlaps with X_{w+1} on samples $x_1, x_2, \cdots, x_{n-1}$. Suppose that the FFT of X_w has been computed. We slide *PHY Win* from X_w to X_{w+1} and aim to obtain the FFT of samples at the new window. Rather than performing another costly FFT at X_{w+1}, we can exploit the data similarity between X_{w+1} and X_w, and directly derive the FFT of X_{w+1} from the obtained FFT results of X_w as below [6]:

$$FFT_i(X_{w+1}) = (FFT_i(X_w) + x_n - x_0)e^{j2\pi i/n}, \tag{9}$$

where $FFT_i(\cdot)$ represents the i'th element of the FFT results, $i = 0, 1, \cdots, n-1$. In Eq. (9), we can obtain the FFT of a new window by updating the FFT of the previous window.

In the context of frequency track extraction, we dechirp and normalize the chirp signals in *PHY Win* before performing FFTs. We combine the dechirping and normalizing operations (*i.e.*, Eqs. (7) and (8)) into one step as below.

$$F^w_{norm} = X_w \cdot C^{-1}(t) \cdot e^{j2\pi(-\frac{k}{2}t_w)}. \tag{10}$$

Our goal is to get the FFT of F^{w+1}_{norm} from the FFT results of F^w_{norm}. In order to apply the optimized method of sliding FFT (*i.e.*, Eq. (9)), we need to ensure that the elements of F^{w+1}_{norm} overlap with those of F^w_{norm}.

Denote the n elements of $C^{-1}(t)$ as $[c_0, c_1, \cdots, c_{n-1}]^T$. Note that $C^{-1}(t) \cdot e^{-j2\pi\frac{k}{2}(t_w+1)} = C^{-1}(t+1) \cdot e^{-j2\pi\frac{k}{2}t_w}$. We have $C^{-1}(t) \cdot e^{-j2\pi\frac{k}{2}(t_w+1)} = [c_1, c_2, \cdots, c_{n-1}, c_n]^T$, where $c_n = c_{n-1}e^{-j2\pi\frac{k}{2}}$. Therefore, we can get the following.

$$\begin{aligned} F^{w+1}_{norm} &= X_{w+1} \cdot C^{-1}(t) \cdot e^{-j2\pi\frac{k}{2}(t_w+1)} \\ &= X_{w+1} \cdot C^{-1}(t+1) \cdot e^{j2\pi(-\frac{k}{2}t_w)} \\ &= [x_1c_1, x_2c_2, \cdots, x_{n-1}c_{n-1}, x_nc_n]^T e^{j2\pi(-\frac{k}{2}t_w)}. \end{aligned}$$

Recall that $F^w_{norm} = [x_0c_0, x_1c_1, \cdots, x_{n-1}c_{n-1}]^T e^{j2\pi(-\frac{k}{2}t_w)}$. F^{w+1}_{norm} overlaps with F^w_{norm} on elements $x_1c_1, x_2c_2, \cdots,$

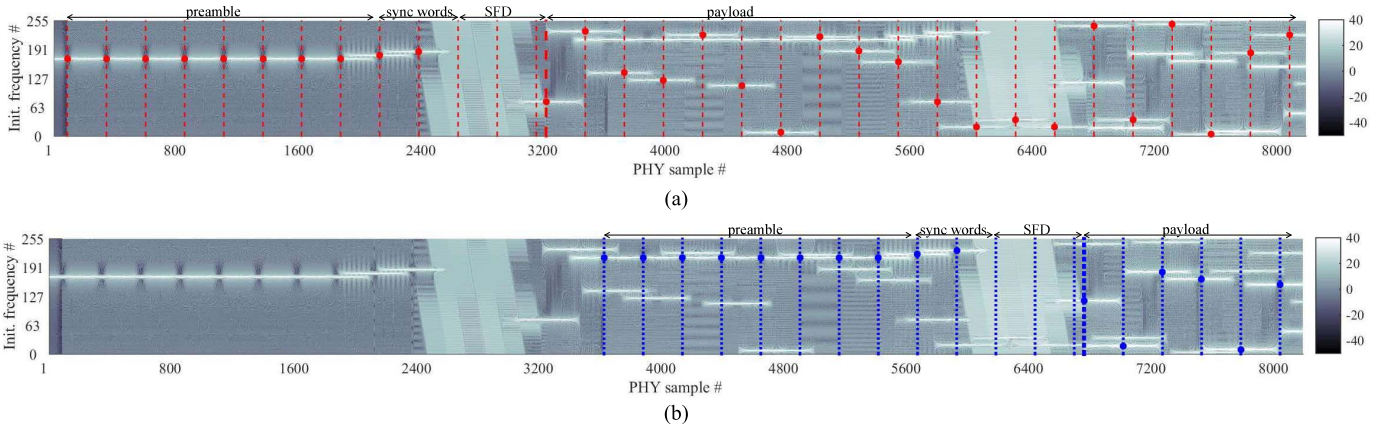


Fig. 13. Separating the signals of Fig. 1 into two LoRa frames: the frame structure and demodulated symbols of (a) the first and (b) second transmission.

$x_{n-1}c_{n-1}$. Therefore, by applying Eq. (9), we obtain

$$FFT_i(F^{w+1}_{norm}) = [FFT_i(F^{w}_{norm}) + x_n c_n e^{j2\pi(-\frac{k}{2}t_w)} - x_0 c_0 e^{j2\pi(-\frac{k}{2}t_w)}] \cdot e^{j2\pi i/n}. \quad (11)$$

This allows us to compute the FFT of F^{w+1}_{norm} from the obtained FFT results of F^{w}_{norm}. We employ Eq. (11) to optimize the computation of sliding FFT. It reduces the time complexity of frequency track extraction to $O(N_s n_{fft})$.

D. FTrack: Put It Together

Fig. 12 illustrates the general workflow of *FTrack*. Specifically, it involves the following key operations:

Frequency track extraction. FTrack first dechirps the received signal with a sliding window. The collided symbols in each window are normalized into the initial frequency of corresponding LoRa chirps (*i.e.*, frequency track). The collided signals are thus converted into a sequence of frequency tracks for further processing.

Frame identification. FTrack extracts long frequency tracks to separate preamble from collisions. It detects symbol edges from the preamble and searches for sync words to identify LoRa packets. The edges of payload symbols can then be deduced from the symbol edges of preamble based on the frame structure of LoRa packets.

Symbol demodulation. FTrack employs the detected symbol edges of a specific preamble to locate the demodulation window of each payload symbol associating with the preamble. FTrack checks the continuity of frequency tracks, as well as the timing information, in each window to filter interference and demodulate symbol from the resulting frequency track of target based on Eq. (8).

Iteration to decode parallel transmissions. FTrack iterates to detect more preambles and demodulate concurrent LoRa transmissions. As collided transmissions may have misaligned symbol edges, they would have different sequences of payload symbol. For example, when we apply FTrack to demodulate the collided LoRa signals shown in Fig. 1, we can detect two LoRa preambles (exhibiting as long frequency tracks), as shown in Fig. 13. Thereafter, we separate the collided symbols into two concurrent transmissions (*i.e.*, frequency track classifying) and demodulate symbols of each transmission. FTrack removes a frequency track after demodulating symbol corresponding to the frequency track, and outputs a sequence of demodulated symbols for each LoRa transmission, as shown in Fig. 13. A conventional LoRa decoder can be applied to decode data from such symbols.

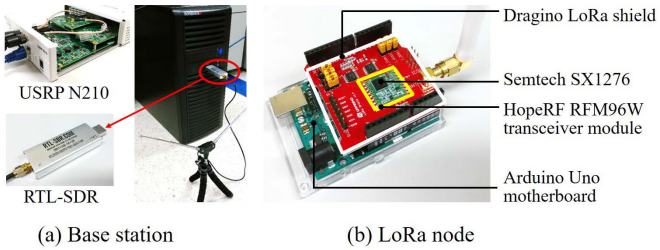


Fig. 14. Experiment equipment.

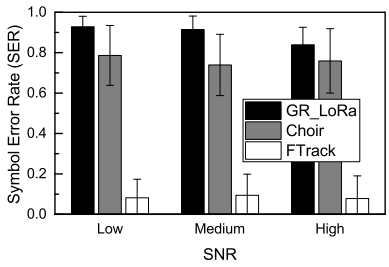
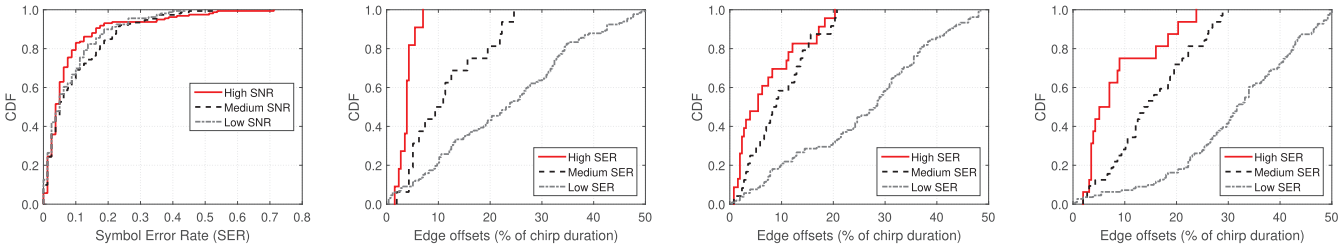


Fig. 15. Demodulation with two-node collisions.

IV. Evaluations

We implement and evaluate the system using software radio base stations and commodity LoRa devices. For performance evaluation, we use both high-end software defined radio (*i.e.*, USRP N210) and low-cost receive-only software defined radio (*i.e.*, RTL-SDR dongle) as shown in Fig. 14(a). We develop our own LoRa demodulator based on the GNU Radio library, and implement FTrack in MATLAB to process PHY samples. If not otherwise specified, we employ RTL-SDR dongles to receive PHY samples at the 900 MHz bands. The USRP N210 is only used for performance evaluations.

The LoRa nodes are composed of Dragino LoRa shields [7], which consist of HopeRF's RFM96W transceiver module embedded with the Semtech SX1276 chip, as shown in Fig. 14(b). We connect the LoRa shield to Arduino Uno motherboard to control SX1276 chip in packet transmission and reception. The SX1276 chip operates at 915MHz with the bandwidth of 250kHz or 125kHz, depending on the



(a) CDF of SERs (b) CDF of edge offsets (High SNR) (c) CDF of edge offsets (Medium SNR) (d) CDF of edge offsets (Low SNR)

Fig. 16. In-depth study of FTrack performance: (a) CDF of the Symbol Error Rates (SERs); (b-d) Relationships between SERs and the normalized edge offsets of colliding symbols: FTrack produces high SERs when colliding symbols are closely-aligned in time (*i.e.*, with short edge offsets).

configuration of LoRa parameters. We note that radio broadcasting, RFID and P-GSM-900 may also work in the 900MHz frequency band. The measured noise floor is around -90dBm in our experiments. By default, we set the spreading factor (SF), coding rate (CR) and bandwidth (BW) of LoRa communication as 8, 4/5 and 250kHz, respectively. The sampling rate of RTL-SDR dongle is 1 MS/s.

A. Parallel Demodulation

1) Basic Performance: In the following experiment, we use an RTL-SDR dongle as a receiver and three LoRa nodes as transmitters in an indoor environment. To experiment with varied colliding time, we configure one transmitter to send beacons every 2 seconds. Upon receiving such beacons, the other two LoRa nodes reply a 30-Byte data frame, which consists of 61 payload symbols and lasts for about 80ms when SF=8, BW=250kHz. Specifically, we configure the two LoRa nodes to delay for a random period of time before transmission with a maximum delay of 20ms (*i.e.*, the air time of 20 symbols). The experiment setting can result in collisions at different parts of packets (*e.g.*, preambles, sync words and payloads) as reported in [8]. We vary the transmission power of LoRa nodes to evaluate demodulation performance across three SNR regimes: low (<5dB), medium (5-20 dB) and high (>20 dB). For each SNR regime, we collect 500 colliding frames and repeat the experiment 5 times. We compare FTrack against GR_LoRa (*i.e.*, a standard LoRa demodulation scheme) and Choir [5] which represents the state-of-the-art on LoRa collision recovery.

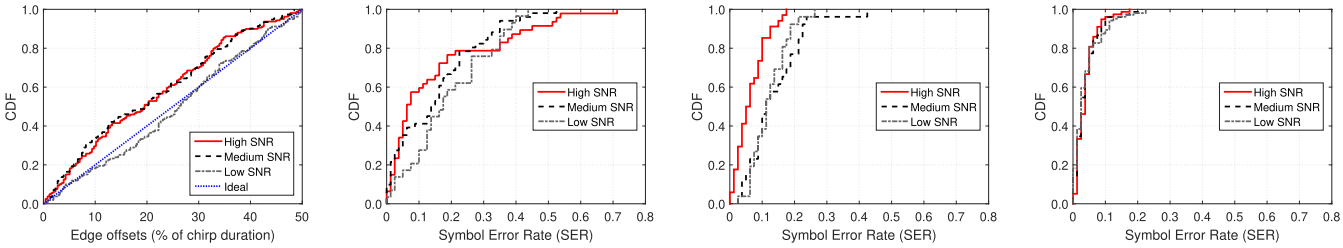
Fig. 15 compares the average SERs of the three demodulation schemes when two LoRa nodes transmit concurrently. According to the experiment results, FTrack performs the best and GR_LoRa is the worst (SER > 80%) because GR_LoRa is not capable of recovering any collisions. Our experimental study reveals that the hardware frequency offset (*i.e.*, the fractional part of chirp frequency) extracted by Choir is not reliable to classify colliding symbols in practice. The main reason is that along with the hardware imperfection, various influencing factors (*e.g.*, phase jitters, noise) may cause variations in the measurement of fractional part of carrier frequency. We also notice that the time offset between a detection window and a symbol edge also influences the measurement of fractional part of carrier frequency. As a result, Choir suffers high symbol error rates (70%∼80%) in our experiments. In contrast, FTrack can leverage the timing misalignment to separate colliding symbols. It yields <10% symbol error rates across all SNR regimes. The low symbol error rates of FTrack can be corrected by standard error correction schemes (*e.g.*, Hamming code) adopted by current LoRa nodes.

We present the CDF of SERs of FTrack in Fig. 16(a). As we can see, FTrack achieves better performance in the high SNR condition than in the low SNR condition. More importantly, the performance of FTrack does not degrade dramatically even in the low SNR condition. That is because FTrack can still leverage the long transmission time of a LoRa packet to boost the signal strength. In particular, 80% of the symbol error rates are below 10% when SNR is high. Even in the medium and low SNR conditions, nearly 70% of the collided frames are demodulated with <10% SERs.

2) Impact of Time Offsets and SNR: We analyze the major factors that may influence the performance of FTrack. Figures 16(b-d) examine the relationships between SERs and the timing offsets of collided symbols. In each SNR regime, we divide the demodulation results into three groups for analysis: low SER (<0.1), medium SER (0.1∼0.2) and high SER (>0.2). We observe that in the high-SNR condition, 100% of high SERs and 50% of medium SERs appear when the edges of colliding symbols are closer than 10% of a chirp duration, as shown in Fig. 16(b). Similar results are observed in the medium and low SNR cases, as shown in Fig.s 16(c,d). That is because once the symbols are not separated in time with sufficient margin, FTrack may not be able to separate such edges, leading to incorrect symbol grouping and demodulating results.

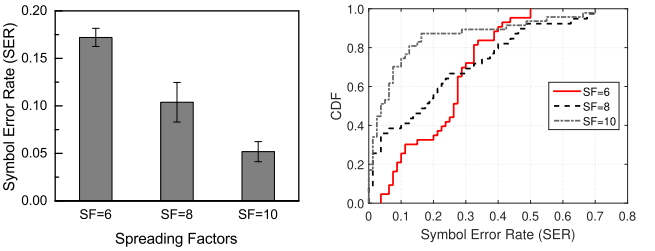
In practice, LoRa nodes may transmit packets in random time slots. Fig. 17(a) shows the edge offsets of collided symbols in our collected traces, where edge offsets are normalized in percentages of a chirp duration. As LoRa nodes transmit at random time, the edge offsets of colliding symbols distribute uniformly across the whole range of one chirp duration (*i.e.*, length of a LoRa symbol). As we mentioned, FTrack may fail to demodulate concurrent transmissions when the edge offsets are short (*e.g.*, <10% chirp duration), while FTrack can successfully recover collisions in most other cases.

We examine the symbol error rates of demodulation when the edge offsets of colliding symbols are small (<10%), medium (10%∼20%) and large (>20%), respectively. By comparing Fig.s 17(b-d), we see that FTrack achieves lower



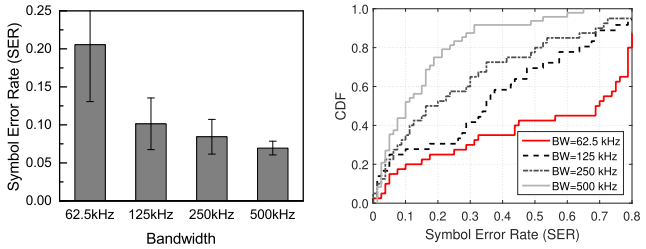
(a) CDF of edge offsets (b) Small edge offsets (<10%) (c) Medium edge offsets (10%∼20%) (d) Large edge offsets (>20%)

Fig. 17. Relationship between the demodulation performance of FTrack and edge offsets of colliding symbols: (a) CDF of edge offsets in the collected frames, (b-d) Demodulation performance when edge offsets are small (<10%), medium (10%–20%) and large (>20%), respectively.



(a) Overall SER performance (b) SERs of small edge offsets (<10%)

Fig. 18. Impact of the Spreading Factor (SF) of LoRa PHY modulation, BW=250kHz: (a) SERs under different SFs, (b) Demodulation results when edge offsets are small (<10%).



(a) Overall SER performance (b) SERs of small edge offsets (<10%)

Fig. 19. Impact of LoRa Bandwidth (BW), SF=8: (a) SERs under different BW settings, (b) Demodulation results when edge offsets are small (<10%).

SERs as the edge offsets of interfering symbols increases. For instance, as shown in Fig. 17(b), when the edge offsets are shorter than 10% chirp duration, nearly 40% of the results of high-SNR (60% of medium-SNR and 80% of low-SNR) have high or medium symbol error rates (*i.e.*, SER>0.1, collisions may not be recovered). This number decreases to lower than 5% when the edge offsets increases to 20%, as shown in Fig. 17(d). In this case, 80% of the results have SERs<0.05.

In addition, by increasing the SNR of received signals, FTrack can produce better demodulation results. For example, as shown in Fig.s 16(b) and (d), when the SNR is low, 82% of low SERs are produced in the case that symbol edges are apart farther than 20% of the symbol length; whereas the results of low SER appear uniformly in all edge offset occasions when SNR becomes high. Therefore, as the SNR increases, FTrack may demodulate with low SERs even when the edges of colliding symbols are closely located. This implies that we can increase the transmission power of LoRa nodes for better collision recovery performance.

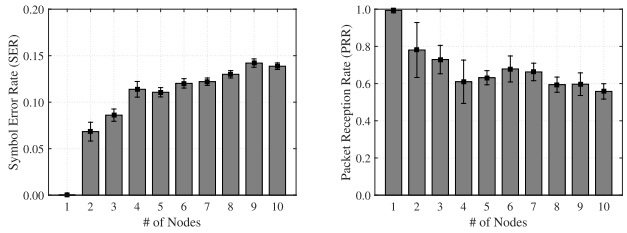
3) Impact of LoRa Packet Configuration: The demodulation performance of FTrack can be affected by the symbol duration of LoRa packets. Some parameters configure the symbol duration of LoRa (*e.g.*, Spreading Factor (SF) and Bandwidth (BW)). In the following, we investigate the impact of SF and BW on collision recovery performance. Unless otherwise specified, we adopt the same experimental settings as in §IV-A.1. We only present the evaluation results of high-SNR. The experiment results exhibit similar trends in both medium-SNR and low-SNR regimes (not presented).

We set the Bandwidth of LoRa to 250kHz and study the impact of Spreading Factor on demodulation performance. Fig. 18(a) presents the overall symbol error rates of FTrack with varied SFs (*e.g.*, SF=6, 8, and 10). We see that FTrack achieves lower SERs with a larger SF. The average SER decreases from 0.17 to 0.05 as SF increases from 6 to 10. In particular, a larger SF facilitates collision recovery especially when the edge offsets of interfering symbols are small, as shown in Fig. 18(b). When SF=6, only 20% of the collisions with small edge offset can be correctly recovered (*i.e.*, SER<0.1). This number increases to 40% for SF=8 and around 70% for SF=10. That is because when SF increases, a symbol takes a longer air time [4], which helps FTrack separate collisions in time.

Next, Fig. 19 evaluates the demodulation performance of FTrack with varied BWs. The Spreading Factor is fixed to 8. The results show that FTrack performs better with a larger BW. The average SER decreases from 0.21 to 0.07 as the LoRa bandwidth increases from 62.5kHz to 500kHz. That is because when the LoRa bandwidth increases, the frequency gap between the symbols coexisting within a demodulation window also increases, which helps the differentiation of symbols in the frequency domain. As shown in Fig. 19(b), when the edge offsets of colliding symbols are small, even though it is hard to separate symbols in time, FTrack recovers more collisions with larger BWs. For instance, less than 20% collisions can be correctly recovered (*i.e.*, SER<0.1) when BW=62.5kHz. The number increases to about 45% when BW=500kHz. However, we note that a larger BW comes at the cost of an increased number of PHY samples and higher computational overhead to process the PHY samples.

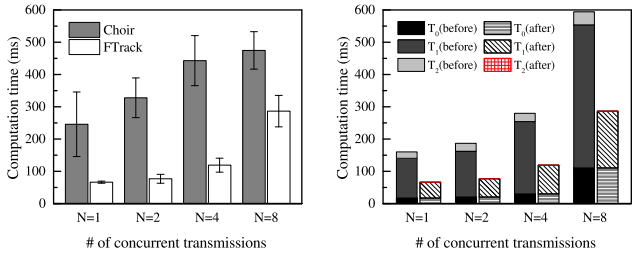
B. FTrack Capability

We examine the capability of FTrack on collision recovery with an increasing number of colliding nodes. In this experiment, we use the same configuration as in §IV-A.1 and



(a) Symbol Error Rate (SER) (b) Packet Reception Ratio (PRR)

Fig. 20. Performance of FTrack on decoding a varied number of parallel transmissions.



(a) Choir vs. FTrack (b) Time costs of FTrack before optimizing vs. after optimizing

Fig. 21. Computational time of collision resolving with various number of concurrent transmissions: (a) The overall time costs (in milliseconds), (b) Anatomy of the time costs of FTrack: T_1—time of frequency track extraction, T_2—frame identification (preamble & symbol edge detecting) and T_0—others (including symbol classifying & iterative frame demodulation).

increase the number of concurrent transmissions from 1 node to 10 nodes. We measure the PHY-layer symbol error rate (SER) and the ratio of packets being correctly decoded (*i.e.*, packet reception ratio, PRR). As shown in Fig. 20, the demodulation errors of FTrack increase as more nodes transmit in parallel. For instance, the average SER is 7% when two nodes collide. Yet it increases to 14% in the case of 10-node collisions. When two nodes transmit concurrently, nearly 80% of the packets can be successfully recovered. The packet reception ratio, however, decreases as more nodes collide. When 10 nodes transmit in parallel, although we cannot recover all packets, 58% of the packets can still be correctly decoded leading to a considerable throughput gain.

C. Real Time Performance

In this experiment, we evaluate the time costs of collision recovery. We record the computation time of Choir and FTrack when running on a PC with the Intel Core i5 CPU processor.Fig. 21(a) compares the overall computation time of collision resolving under different number of concurrent transmissions. Generally, the computational time of FTrack is shorter than that of Choir. The decoding time increases proportionally with the number of collided packets. As the data rates of LoRa are much lower than other wireless technologies (*e.g.*, WiFi, RFID, cellular, *etc.*), it leaves sufficient time for LoRa base stations to decode collisions.

Fig. 21(b) examines the effectiveness of proposed optimizations for FTrack, *i.e.*, optimized edge detecting (§ III-A) and frequency track extraction (§ III-C). We anatomize the time costs of FTrack into three main parts: frequency track extraction (T_1), edge detecting (T_2) and others (T_0). As we can see,

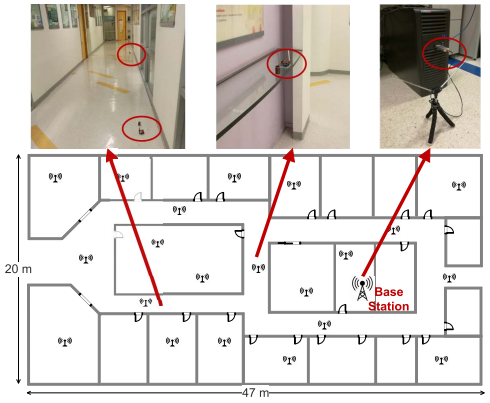


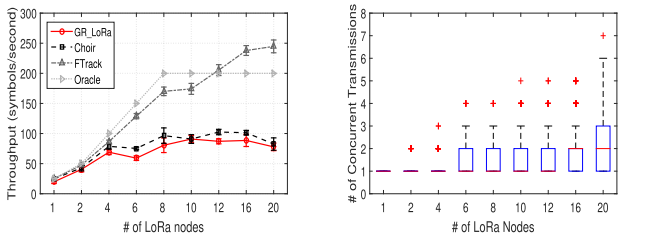
Fig. 22. Layout of a LoRaWAN testbed.

frequency track extraction dominates the computation time of FTrack before optimizing. The optimized method significantly reduces the time costs of frequency track extraction by one order of magnitude, from hundreds of milliseconds to a few tens of milliseconds. The time costs of edge detecting decrease from milliseconds to micro-seconds (*i.e.*, reduced by 1,000×). The overall computation time is reduced by 50%∼60% with the optimized designs of FTrack algorithm. In practice, we can adopt more efficient hardware (*e.g.*, multi-processor, FPGA, ASIC) to further accelerate FTrack so as to meet the real-time processing requirement.

D. Performance in Real Network

In the following, we evaluate how FTrack performs in practice. We deploy the testbed LoRaWAN network within our office building, as illustrated in Fig. 22. Each node senses the environments (*e.g.*, light, temperature, humidity, *etc.*) and randomly selects a 200ms-slot to transmit data to a base station. The payload is 30 Bytes, corresponding to about 80 PHY symbols (payloads + headers) and 80ms air time when SF=8, BW=250kHz. The duty cycle of LoRa nodes are set to 10%, far higher than the typical LoRaWAN configurations of ≤2%, to emulate a serious collision scenario. In particular, the ordinary LoRa nodes in our testbed can transmit in 8 uplink slots during each data collection cycle (the rest two slots are reserved by the LoRaWAN testbed to send control messages). We increase the number of nodes from 1 to 20 to compare the scalability of different approaches. We compare FTrack with GR_LoRa [9] and Choir [5]. We also compare the performance against an Oracle scheme that is assumed to optimally schedule the LoRa nodes such that no collision could happen.

Fig. 23(a) shows the network throughput of four approaches. We see that the throughput of all approaches increase as more nodes join the network, when the network size is small (*e.g.*, # of nodes≤4). However, the throughput of GR_LoRa and Choir saturate (around 80 symbols/sec) rapidly when the network size increases to 4 nodes. That is because as more nodes transmit concurrently, GR_LoRa and Choir cannot recover collisions among the nodes. Benefiting from a perfect transmission schedule, Oracle yields the highest throughput that increases linearly with network size as the number of nodes increases from 1 to 8. However, the Oracle reaches its capacity



(a) Network throughput (b) Number of concurrent LoRa transmissions

Fig. 23. Performance of decoding concurrent transmissions in real-world LoRaWAN network (SF=8, BW=250kHz).

limit (200 symbols/sec) when network size is 8. In contrast, the throughput of FTrack further increases to 250 symbols/second as network size increases to 20 nodes, which is about 3 times the throughput of Choir and GR_LoRa. The performance gain stems from FTrack's capability of decoding concurrent LoRa transmissions.

Fig. 23(b) presents the number of concurrent transmissions with a varied number of LoRa nodes working in the low-duty cycle mode. We see that when the network size exceeds 4, nodes collide more frequently even if they work in the low-duty cycle mode. As such, the performance of GR_LoRa and Choir starts to degrade when the number of nodes exceeds 4, as shown in Fig. 23(a). As the network size further increases, we observe more collisions: A median of 2 and a maximum of 5 nodes transmit concurrently at a slot when the network size exceeds 16. In this case, FTrack achieves the highest network throughput as it is able to recover collisions.

V. Related Work

A variety of LP-WAN technologies [10], [11] have been proposed to enable the fast-growing IoT applications. SigFox [1] uses Ultra Narrow Band (UNB) technology combined with DBPSK and GFSK modulation to support low-power long-range communication in the ISM band. NB-IoT [2] and LTE-M [12] are introduced by 3GPP. They use a subset of the LTE standard, operate at licensed spectrum yet limit the bandwidth to a single narrow-band of 200kHz. LoRa [3], [11], like SigFox, works at license-free frequency band. It employs Chirp Spread Spectrum (CSS) modulation to transmit data, which is resilient to interference, multi-path fading and Doppler effects. Such characteristics make LoRa a competitive candidate for long-range low-cost IoT networks [13], [14]. We focus on LoRa in this article and refer the readers to [15]–[17] for detailed comparisons of existing LP-WAN technologies.

There are some pioneer researches on LoRa and LoRaWAN [18], [19]. Early efforts have been devoted to the measurement study [4], [20], [21] and performance analysis, such as transmission air time [22], [23], power consumption [4], [24]–[26], coverage [27], [28], physical layer security [29], *etc*. Based on these studies, some improvement schemes [30]–[32] are introduced for better performance. Although the LoRa PHY is proprietary, authors of [9] and [33] employ reverse engineering to study the encoding and decoding schemes of LoRa.

With respect to the limited capability of hardware, LoRaWAN [3] adopts a simple ALOHA-based MAC for access control. The limits of LoRaWAN MAC have been analytically concluded in [34], [35]. [36] studies LoRa collisions via simulation. [37] employs commodity devices to empirically study the characteristic of LoRa collisions. More recently, [8] presents an in-depth investigation of LoRa collisions within actual running networks. To avoid collisions, some researchers [38], [39] proposed new MAC designs that incorporate advanced scheduling schemes on top of LoRa. However, these schemes would add higher complexity yet produce lower efficiency in terms of network capacity and throughput.

The work most related to ours is Choir [5], which exploits the frequency offsets introduced by LoRa hardware to separate collisions. However, in practice, the extracted frequency offset is not reliable to classify colliding symbols due to various influencing factors (*e.g.*, phase jitters, time offset).

Our work [40] is inspired by the previous works of collision recovery and parallel decoding in various wireless systems (*e.g.*, WiFi, RFID). ZigZag [41] decodes collisions by exploiting the fact that the time offset of collided transmissions produces some interference-free chunks. It extracts the interference-free chunks and subtracts from collisions to separate each individual transmission. BiGroup [42] examines the collision states of concurrent transmissions of RFID tags, and iteratively detects the state transitions of collided signals to decode tag transmissions. LF-Backscatter [43] employs powerful RFID readers to detect the interleaving signal edges with high sampling rates and separate collided tag signals. More recently, FlipTracer [44] and Hubble [45] support parallel decoding of backscatter communications by leveraging both PHY and time domain information. NetScatter [46] proposes a new coding scheme that combines On-Off Keying and CSS to support concurrent transmissions. Recently, [47] and [48] also studied the problem of LoRa collision resolving by using specific features of LoRa chirps in the frequency and time domain. They are complimentary to FTrack.

In addition to the collision recovery approach, some works aim to *avoid collisions*, including MIMO [49]–[51], TDMA [52], [53], collision-recovery methods [54], [55], and constructive interference [56], [57]. LoRa also supports orthogonal communications by transmitting with different channels, bandwidths and spreading factors. However, even with orthogonal parameter settings, transmission collisions may still happen as the number of devices further increases, *e.g.*, in scenarios like urban and warehouse [8], [37]. Our work is complementary to such collision avoidance schemes.

VI. Discussions

Demodulating capacity and the gain. The demodulation capacity of FTrack is not unbounded as the number of colliding nodes increases. If the edges of collided symbols are aligned or closely-located, FTrack may not be able to separate them in time. Besides, the configurations of LoRa transmission, such as spreading factor (SF) and bandwidth (BW), also affect the demodulation performance. In fact, current

LoRaWAN supports concurrent transmission of packets using orthogonal spreading factors which require careful configuration and coordinations of LoRa nodes. However, due to lack of coordination, some nodes may still collide if they choose the same parameter settings especially in densely-populated LoRa networks (*e.g.*, an urban or warehouse scenario). As such, the proposed collision recovery strategy can be complementary to current LoRa network.

Accounting for SNR variations and the near-far problem. In practice, FTrack may fail to detect the frequency power of weak transmitter (*i.e.*, false negative error) or mistakenly detect the power leaking from the main frequency of strong transmitter as frequency track (*i.e.*, false positive error). We solve the problem by selecting thresholds for frequency track detection dynamically based on the SNR conditions. Moreover, to handle the case that strong receptions of a nearby transmitter overwhelm the weak signals of far-away transmitters (*i.e.*, the near-far problem [58]), we can employ a method similar to ZigZag [41] and successive interference cancellation [59] to extract signals of comparable power strength. We apply FTrack on the signals with similar power strength to detect frequency tracks.

Combating the absence of LoRa configurations. Generally, the BW and SF configurations are chosen by LoRa transmitter. A receiver may not know the configurations of received frames. With FTrack, we can detect BW and SF from a short segment of the received signals (*e.g.*, $\leq$ 5 chirps). We exploit the fact that chirps are transformed into horizontal frequency tracks only if down-chirp $C^{-1}(t)$ is produced with the correct BW and SF. One approach is to detect the existence of frequency tracks by trying all BW-SF combinations. To accelerate the process, we can process the PHY samples in parallel with multiple threads, each of which is configured by a particular pair of BW and SF.

Limitations. In practice, some LoRaWAN MAC protocols schedule LoRa transmissions with CSMA or random slotting mechanisms, which can lead to collisions in synchronized time slots. As a result, the symbol edges of collided frames can be aligned, which affects the capability of FTrack on collision recovery. To address this problem, we can modify the MAC protocols in practice to intentionally mis-align collided transmissions. Moreover, we can properly select the parameters of LoRa communication (*e.g.*, spreading factor, bandwidth, transmission power, *etc*.) to maximize the capability of FTrack in supporting more parallel transmissions.

VII. Conclusion

This article presents FTrack, a practical strategy that resolves LoRa collisions in both time and frequency domains. FTrack jointly exploits the distinct frequency tracks and misaligned edges of LoRa symbols to separate collisions. It enables a novel communication paradigm that allows LoRa node to join on-going communications in parallel without specific coordination. We optimize FTrack for real-time decoding and implement it on a low-cost SDR platform. We deploy an indoor testbed to evaluate the performance of FTrack in a variety of network settings. Results show that FTrack recovers collided LoRa frames with low symbol error rates in diverse SNR conditions. It can boost the throughput of real-world LoRaWAN by up to 3 times. The parallel decoding capability of FTrack can benefit the deployment of large-scale LoRaWAN in densely-populated IoT scenarios.

References

[1] SigFox. (Jan. 2019). *SigFox Overview*. [Online]. Available: https://www.sigfox.com/en/sigfox-iot-technology-overview

[2] R. Ratasuk, B. Vejlgaard, N. Mangalvedhe, and A. Ghosh, "NB-IoT system for M2M communication," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2016, pp. 1–5.

[3] LoRa Alliance. (Jan. 2019). *Lorawan for Developer*. [Online]. Available: https://lora-alliance.org/lorawan-for-developers

[4] J. Liando, A. Gamage, A. Tengourtius, and M. Li, "Known and unkown facts of LoRa: Experiences from a large scale measurement study," *ACM Trans. Sensor Netw.*, vol. 15, no. 2, pp. 16:1–16:35, Feb. 2019.

[5] 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. (SIGCOMM)*, Aug. 2017, pp. 309–321.

[6] E. Jacobsen and R. Lyons, "The sliding DFT," *IEEE Signal Process. Mag.*, vol. 20, no. 2, pp. 74–80, Mar. 2003.

[7] Dragino. (Mar. 2019). *LoRa Shield for Arduino*. [Online]. Available: http://www.dragino.com/products/module/item/102-lora-shield.html

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

[9] M. Knight and B. Seeber, "Decoding LoRa: Realizing a modern LPWAN with SDR," in *Proc. 6th GNU Radio Conf.*, Sep. 2016, pp. 1–5.

[10] A. Lavric and A. I. Petrariu, "LoRaWAN communication protocol: The new era of IoT," in *Proc. Int. Conf. Develop. Appl. Syst. (DAS)*, May 2018, pp. 74–77.

[11] J.-P. Bardyn, T. Melly, O. Seller, and N. Sornin, "IoT: The era of LPWAN is starting now," in *Proc. 42nd Eur. Solid-State Circuits Conf. (ESSCIRC)*, Sep. 2016, pp. 25–30.

[12] M. Lauridsen, I. Z. Kovacs, P. Mogensen, M. Sorensen, and S. Holst, "Coverage and capacity analysis of LTE-M and NB-IoT in a rural area," in *Proc. IEEE 84th Veh. Technol. Conf. (VTC-Fall)*, Sep. 2016, pp. 1–5.

[13] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley, "A study of LoRa: Long range & low power networks for the Internet of Things," *Sensors*, vol. 16, no. 9, p. 1446, Sep. 2016.

[14] J. de Carvalho Silva, J. J. Rodrigues, A. M. Alberti, P. Solic, and A. L. 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.

[15] R. S. Sinha, Y. Wei, and S.-H. Hwang, "A survey on LPWA technology: LoRa and NB-IoT," *ICT Express*, vol. 3, no. 1, pp. 14–21, Mar. 2017.

[16] B. Vejlgaard, M. Lauridsen, H. Nguyen, I. Z. Kovacs, P. Mogensen, and M. Sorensen, "Coverage and capacity analysis of sigfox, LoRa, GPRS, and NB-IoT," in *Proc. IEEE 85th Veh. Technol. Conf. (VTC Spring)*, Jun. 2017, pp. 1–5.

[17] J. P. S. Sundaram, W. Du, and Z. Zhao, "A survey on LoRa networking: Research problems, current solutions, and open issues," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 371–388, 1st Quart., 2020.

[18] J. Haxhibeqiri, E. De Poorter, I. Moerman, and J. Hoebeke, "A survey of LoRaWAN for IoT: From technology to application," *Sensors*, vol. 18, no. 11, p. 3995, Nov. 2018.

[19] A. Gamage, J. C. Liando, C. Gu, R. Tan, and M. Li, "LMAC: Efficient carrier-sense multiple access for lora," in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, to be published.

[20] N. Blenn and F. A. Kuipers, "LoRaWAN in the wild: Measurements from the things network," *CoRR*, 2017, pp. 1–9.

[21] A. Carlsson, I. Kuzminykh, R. Franksson, and A. Liljegren, "Measuring a LoRa network: Performance, possibilities and limitations," in *Proc. 18th Int. Conf. NEWAN, 11th Conf. ruSMART*, Aug. 2018, pp. 116–128.

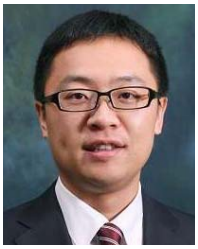
[22] U. Noreen, A. Bounceur, and L. Clavier, "A study of LoRa low power and wide area network technology," in *Proc. Int. Conf. Adv. Technol. Signal Image Process. (ATSIP)*, Oct. 2017, pp. 1–6.

[23] A. Lavric and V. Popa, "A LoRaWAN: Long range wide area networks study," in *Proc. Int. Conf. Electromech. Power Syst. (SIELMEN)*, Oct. 2017, pp. 417–420.

[24] T. Bouguera, J.-F. Diouris, J.-J. Chaillout, R. Jaouadi, and G. Andrieux, "Energy consumption model for sensor nodes based on LoRa and LoRaWAN," *Sensors*, vol. 18, no. 7, p. 2104, Jun. 2018.

[25] 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.
[26] X. Xia, Y. Zheng, and T. Gu, "LiteNap: Downclocking LoRa reception," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Jul. 2020, pp. 2321–2330.
[27] J. Petajajarvi, K. Mikhaylov, A. Roivainen, T. Hanninen, and M. Pettissalo, "On the coverage of LPWANs: Range evaluation and channel attenuation model for LoRa technology," in *Proc. 14th Int. Conf. ITS Telecommun. (ITST)*, Dec. 2015, pp. 55–59.
[28] J. Petäjäjärvi, K. Mikhaylov, M. Pettissalo, J. Janhunen, and J. Iinatti, "Performance of a low-power wide-area network based on LoRa technology: Doppler robustness, scalability, and coverage," *Int. J. Distrib. Sensor Netw.*, vol. 13, no. 3, pp. 1–16, Mar. 2017.
[29] N. Hou and Y. Zheng, "Cloaklora: A covert channel over LoRa phy," in *Proc. 28th IEEE Int. Conf. Netw. Protocols (ICNP)*, Oct. 2020, pp. 1–9.
[30] B. Reynders, W. Meert, and S. Pollin, "Power and spreading factor control in low power wide area networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.
[31] J.-T. Lim and Y. Han, "Spreading factor allocation for massive connectivity in LoRa systems," *IEEE Commun. Lett.*, vol. 22, no. 4, pp. 800–803, Apr. 2018.
[32] K. Abdelfadeel, V. Cionca, and D. Pesch, "Poster: A fair adaptive data rate algorithm for LoRaWAN," in *Proc. Int. Conf. Embedded Wireless Syst. Netw. (EWSN)*, 2018, pp. 169–170.
[33] P. Robyns, P. Quax, W. Lamotte, and W. Thenaers, "A multi-channel software decoder for the LoRa modulation scheme," in *Proc. 3rd Int. Conf. Internet Things, Big Data Secur. (IoTBDS)*, 2018, pp. 1–11.
[34] 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.
[35] D. Bankov, E. Khorov, and A. Lyakhov, "On the limits of LoRaWAN channel access," in *Proc. Int. Conf. Eng. Telecommun. (EnT)*, Nov. 2016, pp. 10–14.
[36] M. C. Bor, U. Roedig, T. Voigt, and J. M. Alonso, "Do LoRa low-power wide-area networks scale?" in *Proc. 19th ACM Int. Conf. Modeling, Anal. Simulation Wireless Mobile Syst. (MSWiM)*, Nov. 2016, pp. 59–67.
[37] J. Haxhibeqiri, F. Van den Abeele, I. Moerman, and J. Hoebeke, "LoRa scalability: A simulation model based on interference measurements," *Sensors*, vol. 17, no. 6, p. 1193, May 2017.
[38] B. Reynders, Q. Wang, P. Tuset-Peiro, X. Vilajosana, and S. Pollin, "Improving reliability and scalability of LoRaWANs through lightweight scheduling," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1830–1842, Jun. 2018.
[39] C. Pham, "Investigating and experimenting CSMA channel access mechanisms for LoRa IoT networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2018, pp. 1–6.
[40] X. Xia, Y. Zheng, and T. Gu, "FTrack: Parallel decoding for LoRa transmissions," in *Proc. 17th Conf. Embedded Netw. Sensor Syst. (SenSys)*, Nov. 2019, pp. 192–204.
[41] S. Gollakota and D. Katabi, "Zigzag decoding: Combating hidden terminals in wireless networks," in *Proc. ACM SIGCOMM Conf. Data Commun. (SIGCOMM)*, Aug. 2008, pp. 159–170.
[42] J. Ou, M. Li, and Y. Zheng, "Come and be served: Parallel decoding for cots RFID tags," in *Proc. 21st Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, Sep. 2015, pp. 500–511.
[43] P. Hu, P. Zhang, and D. Ganesan, "Laissez-faire: Fully asymmetric backscatter communication," in *Proc. ACM Conf. Special Interest Group Data Commun. (SIGCOMM)*, Aug. 2015, pp. 255–267.
[44] M. Jin, Y. He, X. Meng, Y. Zheng, D. Fang, and X. Chen, "Fliptracer: Practical parallel decoding for backscatter communication," in *Proc. 23rd Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2017, pp. 275–287.
[45] M. Jin, Y. He, X. Meng, D. Fang, and X. Chen, "Parallel backscatter in the wild: When burstiness and randomness play with you," in *Proc. 24th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2018, pp. 471–485.
[46] M. Hessar, A. Najafi, and S. Gollakota, "NetScatter: Enabling large-scale backscatter networks," in *Proc. 16th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2019, pp. 271–284.
[47] 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.
[48] 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 (MobiSys)*, Jun. 2020, pp. 234–246.
[49] N. Anand, R. E. Guerra, and E. W. Knightly, "The case for UHF-band MU-MIMO," in *Proc. 20th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2014, pp. 29–40.
[50] H. S. Rahul, S. Kumar, and D. Katabi, "JMB: Scaling wireless capacity with user demands," in *Proc. ACM SIGCOMM Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, 2012, pp. 235–246.
[51] Y. Xie, J. Xiong, M. Li, and K. Jamieson, "mD-track: Leveraging multi-dimensionality for passive indoor Wi-Fi tracking," in *Proc. MobiCom*, 2019, pp. 8:1–8:16.
[52] A. Rowe, D. Goel, and R. Rajkumar, "FireFly mosaic: A vision-enabled wireless sensor networking system," in *Proc. 28th IEEE Int. Real-Time Syst. Symp. (RTSS)*, Dec. 2007, pp. 459–468.
[53] X. Xia *et al.*, "Surviving screen-off battery through out-of-band Wi-Fi coordination," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.
[54] K. Jamieson and H. Balakrishnan, "PPR: Partial packet recovery for wireless networks," in *Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun. (SIGCOMM)*, 2007, pp. 409–420.
[55] Y. Wu, G. Zhou, and J. A. Stankovic, "ACR: Active collision recovery in dense wireless sensor networks," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.
[56] W. Du, J. C. Liando, H. Zhang, and M. Li, "When pipelines meet fountain: Fast data dissemination in wireless sensor networks," in *Proc. ACM Conf. Embedded Netw. Sensor Syst.*, 2015, pp. 365–378.
[57] Z. Li, W. Du, Y. Zheng, M. Li, and D. Wu, "From rateless to hopless," *IEEE/ACM Trans. Netw.*, vol. 25, no. 1, pp. 69–82, Feb. 2017.
[58] D. J. Goodman and A. A. M. Saleh, "The near/far effect in local ALOHA radio communications," *IEEE Trans. Veh. Technol.*, vol. VT-36, no. 1, pp. 19–27, Feb. 1987.
[59] D. Halperin, T. Anderson, and D. Wetherall, "Taking the sting out of carrier sense: Interference cancellation for wireless LANs," in *Proc. 14th ACM Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2008, pp. 339–350.

Xianjin Xia (Member, IEEE) received the B.S., M.Sc., and Ph.D. degrees in computer science from Northwestern Polytechnical University, Xi'an, China, in 2010, 2013, and 2018, respectively. He is currently a Post-Doctoral Research Fellow with the Department of Computing, The Hong Kong Polytechnic University. His research interests include low-power wide-area networks, localization, mobile computing, and so on. He is a member of the ACM.



Yuanqing Zheng (Member, IEEE) received the B.S. degree in electrical engineering and the M.E. degree in communication and information system from Beijing Normal University, Beijing, China, in 2007 and 2010, respectively, and the Ph.D. degree from the School of Computer Engineering, Nanyang Technological University, in 2014. He is currently an Associate Professor with the Department of Computing, The Hong Kong Polytechnic University. His research interest includes wireless networking and mobile computing, acoustic and RF sensing, and the Internet of Things (IoT). He is a member of the ACM.

Tao Gu (Senior Member, IEEE) received the bachelor's degree from the Huazhong University of Science and Technology, the M.Sc. degree from Nanyang Technological University, Singapore, and the Ph.D. degree in computer science from the National University of Singapore. He is currently an Associate Professor with the School of Computer Science and IT, RMIT University, Australia. His research interests include mobile computing, ubiquitous computing, wireless sensor networks, sensor data analytics, and the Internet of Things. He is a member of the ACM.