Enabling Cross Technology Communication from LR-FHSS to LoRa

Md Ashikul Haque
University of Texas at Dallas
Dallas, USA
mdashikul.haque@utdallas.edu

Aakriti Jain
University of Texas at Dallas
Dallas, USA
Aakriti.Jain@UTDallas.edu

Prashant Modekurthy
University of Nevada, Las Vegas
Las Vegas, USA
prashant.modekurthy@unlv.edu

Abusayeed Saifullah
University of Texas at Dallas
Dallas, USA
Abusayeed.Saifullah@UTDallas.edu

Abstract

LR-FHSS extends LoRa uplink coverage to tens of kilometers, but a fundamental asymmetry persists in practice: LR-FHSS nodes are transmit only, and although they contain LoRa receivers, LoRa downlinks fail far short of LR-FHSS ranges. Our outdoor experiments show that LoRa packet reception rate (PRR) collapses beyond 1–2 km while LR-FHSS maintains more than 79% PRR, making ACK, downlink control, and LR-FHSS to LoRa device communication infeasible at long range. Existing CTC techniques cannot address this gap because LoRa requires precise linear chirps, whereas LR-FHSS emits discrete frequency hopped GMSK bursts with mandatory hop gaps and continuous phase memory. We introduce a CTC technique that repurposes the LR-FHSS physical layer into a waveform synthesizer capable of generating LoRa compatible pseudo chirps using either pure tone emission or continuous phase tones derived from GMSK. At the LoRa receiver, a set of physical layer reconstruction mechanisms correct hop discontinuity, offset drift, and GMSK induced phase distortion so that commodity LoRa hardware can decode the resulting chirps. Implemented on USRP B200 with GNU Radio and evaluated across different outdoor scenarios, our technique improves long range reliability by up to 79% over LoRa and increases end to end LR-FHSS PRR by about 15%, enabling a practical long-range downlink and control path without RF front-end changes, assuming PHY-hook access at both the LR-FHSS gateway and the LoRa receiver.

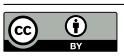
CCS Concepts

• **Computer systems organization** → **Embedded systems**; • **Networks** → *Network reliability*.

Keywords

low-power wide-area networks, LoRa, LR-FHSS, cross-technology communication, wireless IoT

ACM Reference Format:
Md Ashikul Haque, Aakriti Jain, Prashant Modekurthy, and Abusayeed Saifullah. 2026. Enabling Cross Technology Communication from LR-FHSS to LoRa. In *ACM/IEEE International Conference on Embedded Artificial Intelligence and Sensing Systems (SenSys '26), May 11–14, 2026, Saint Malo, France.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3774906.3802771



SenSys '26, Saint Malo, France

ACM ISBN 979-8-4007-2309-4/26/05
https://doi.org/10.1145/3774906.3802771

1 Introduction

Low-Power Wide-Area Networks (LPWANs) have become a foundational technology for the Internet of Things (IoT), enabling connectivity for billions of low-cost devices distributed over vast geographical areas. Among these, LoRa [7–9, 11] has seen widespread adoption, leveraging Chirp Spread Spectrum (CSS) modulation to balance communication range, power efficiency, and interference resilience.

Yet for many long-range sensing applications, such as wildlife and habitat monitoring, open-field grazing, precision agriculture, and direct-to-satellite (DtS) IoT, the communication range of LoRa remains a limiting factor. The Long Range–Frequency Hopping Spread Spectrum (LR-FHSS) [21] physical layer was introduced to overcome this limitation. By transmitting bits of the same packet over ultra-narrow subcarriers (488 Hz) and performing robust frequency hopping, LR-FHSS achieves an exceptional link budget and strong interference resilience, enabling reliable connectivity over tens of kilometers. This makes it ideal for uplink data collection from remote or sparsely deployed sensors.

However, LR-FHSS deployments show a fundamental and often overlooked asymmetry. Commercially available LR-FHSS nodes are *transmit-only*, i.e., these nodes can transmit LR-FHSS packets and cannot receive LR-FHSS packets. LR-FHSS nodes are not equipped with LR-FHSS receivers since they require high power and hardware complexity. Gateways, on the other hand, are equipped with full LR-FHSS transceivers. LR-FHSS nodes and gateways are equipped with LoRa transceivers. In principle, downlink communication can occur over LoRa; however, LoRa and LR-FHSS have different communication ranges. LR-FHSS supports communication over tens of kilometers, but LoRa supports communication over a few kilometers.

Range disparity. To demonstrate the disparity in communication ranges, we conducted an outdoor experiment in an urban non-Line-of-Sight setting to measure the packet reception rate (PRR) of LoRa and LR-FHSS by varying the distance. Figure 1 shows the results of this experiment and highlights the mismatch in communication range between LoRa and LR-FHSS. LoRa's PRR drops from over 70% at 500 m to nearly 0% at 2 km. Under the same conditions, LR-FHSS

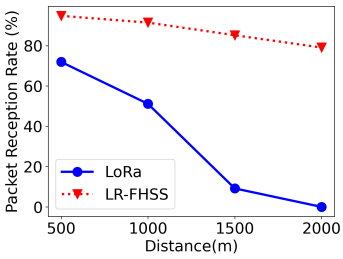


Figure 1: Measured PRR across distance. LoRa fails beyond ∼1–2 km, while LR-FHSS remains robust along the same path. (Configuration: LoRa SF=10, BW=125 kHz; LR-FHSS uses the same transmit power (20 dBm) as LoRa; 100 packets/configuration; antenna height 20 ft above ground; non-line-of-sight.) This asymmetry underpins our goal: enabling LR-FHSS gateways to reach LoRa receivers at long range.

maintains a PRR above 79%. Our experiment results demonstrate that LoRa is not suitable for supporting downlink communication between LR-FHSS gateway and LR-FHSS nodes over tens of kilometers.

Implications. The range disparity between LoRa and LR-FHSS limits the LR-FHSS gateway's abilities. It cannot send ACKs, instruct remote nodes to adapt reporting rates, change sensing modes, initiate emergency transmissions, or coordinate behaviors across nodes, even when their uplinks are received successfully. Thus, the unavailability of LR-FHSS receivers at nodes and range disparity between LoRa and LR-FHSS severely limit the applicability of LR-FHSS. Specifically, LR-FHSS is less suitable for applications that require gateways to send control, coordination, or configuration commands to nodes. It is also less usable in applications that require high reliability.

Motivation for LR FHSS to LoRa CTC. Some use cases require interoperability between LR-FHSS and LoRa networks. For example, LR-FHSS nodes can serve to interconnect disconnected LoRa networks for disseminating emergency information. Additionally, LR-FHSS nodes may leverage LoRa's widespread deployment to establish communication in emergency situations or expand capacity without the need to replace the existing network infrastructure [1]. These applications, in addition to transmitting ACKs and downink control commands from gateway to nodes, motivate the need for a cross-technology-communication (CTC) between LR-FHSS and LoRa.

CTC repurposes one radio technology to generate waveforms interpretable by another without modifying hardware. Prior works on CTC have addressed similar challenges across heterogeneous wireless technologies. Packet level techniques such as FreeBee and C Morse [14, 27] encode bits through timing and energy patterns, while physical level emulation systems such as WEBee, BlueBee, LongBee, LTE2B, XFi, Receiver-Side-CTC, SymBee, and PCTC [12, 13, 16–19, 24, 26] synthesize physical layer (PHY) compatible signals across technologies (for example WiFi to ZigBee or BLE to ZigBee). For LoRa specifically, BLE to LoRa [15] has demonstrated that chirp emulation from a non LoRa PHY is feasible. However, LoRa's CSS requires chirps with linear frequency modulation, whereas LR-FHSS uses discrete frequency hopped GMSK transmissions with strict spectral separation. Bridging these two incompatible PHY layers is highly challenging.

Our vision. In this paper, we present a CTC technique that enables an LR-FHSS gateway to transmit messages that can be decoded by LoRa receivers over the existing RF front-end hardware, assuming PHY hooks at both ends: baseband-level control on the LR-FHSS gateway and access to complex I/Q samples together with a pre-demodulation insertion point on the LoRa receiver. The core idea is to repurpose the LR-FHSS PHY from a data modulator into a flexible waveform synthesizer. By overriding the pseudo-random hopping sequence and emitting a controlled sequence of subcarrier tones, the gateway constructs a "Pseudo-Chirp" that mimics LoRa chirps.

Importantly, our technique incorporates two complementary Pseudo-Chirp synthesis modes consistent with Section 4 and Section 5. In the first mode, the system bypasses GMSK and emits pure tones for maximum chirp fidelity. In the second mode, it preserves GMSK and shapes continuous-phase tones derived from the underlying continuous phase modulation (CPM). This second mode supports deployments where GMSK cannot be bypassed, providing an alternative technique that incurs an average 10% PRR reduction due to CPM memory and bounded frequency deviation. Together, these modes span a practical design trade-off between modulation control and reconstruction complexity.

Challenges and solutions. Synthesizing a LoRa-compatible chirp from LR-FHSS transmissions introduces several challenges absent in prior CTC systems. First, LoRa receiver activation is difficult because the energy distribution of our synthesized preamble, whether generated using pure tones or GMSK, differs from that of a native LoRa preamble, resulting in inconsistent Channel Activity Detection (CAD) triggering. Second, US Federal Communication Commission (FCC) mandates that a frequency hop is separated by a minimum of 25.4 kHz, which fragments the chirp and scatter energy across FFT bins. Third, LR-FHSS's GMSK-based continuous phase modulation introduces nonlinear phase evolution that is inherently misaligned with the ideal linear chirp structure expected by LoRa. Our technique overcomes these through a *PHY-layer architecture*: a specialized activation preamble at the gateway, precise offset correction at the LoRa node, and a **Chirp Reconstruction Module (CRM)** that repairs hop discontinuity and GMSK-induced distortions to produce a clean, continuous chirp for native LoRa demodulation.

Contributions. This paper makes the following contributions:

- We identify and characterize the fundamental downlink asymmetry created by the severe distance gap between LoRa and LR-FHSS, showing why LoRa is unsuitable for long-range control in LR-FHSS deployments.
- We design *the first* CTC technique that synthesizes and reconstructs pseudo-chirps from LR-FHSS transmissions, supporting both pure-tone and GMSK-based synthesis, to achieve LoRa-compatible communication.
- We empirically demonstrate that our Chirp Reconstruction Module transforms fragmented LR-FHSS signals into valid LoRa chirps decodable by commodity LoRa hardware, enabling reliable long-range cross-technology control.

- We implemented our CTC technique on a USRP B200 [2] using GNU Radio [3] and evaluated it across diverse outdoor scenarios. Our results show up to a 79% PRR improvement over native LoRa at long range and a 15% PRR improvement in end-to-end LR-FHSS communication by enabling ACK through our CTC technique.

The rest of this paper is organized as follows. Section 2 reviews the literature in LoRa/LR-FHSS and cross-technology communication. Section 3 summarizes necessary LR-FHSS and LoRa physical-layer concepts. Section 4 introduces the core idea of pseudo-chirp synthesis using LR-FHSS. Section 5 presents the challenges of using LR-FHSS's GMSK modulation for pseudo-chirp synthesis and the techniques we develop to mitigate them. Section 6 outlines the key technical obstacles in realizing our core ideas. Section 7 provides an overview of our technique's PHY-layer architecture at both the transmitter and receiver. Section 8 details the PHY-layer design at the transmitter, and Section 9 details the corresponding design at the receiver. Section 10 evaluates our technique under diverse real-world conditions, and Section 11 concludes the paper.

2 Related Work

Recent developments in CTC have been introduced to establish direct connection across technologies in order to resolve the inherent problems of bridging wireless technologies [4, 5, 10, 13, 14]. At the packet level, techniques such as FreeBee [14] and C-Morse [27] encode information by manipulating packet timing or energy patterns, which can be detected via RSSI or CSI measurements. These methods offer reliable, low-complexity communication with moderate throughput and minimal hardware modification. To achieve higher throughput, physical-layer CTC approaches emulate the transmitter signals of one technology at the PHY layer of another. Examples include WEBee [16], BlueBee [13], LongBee [17], and LTE2B [18], which demonstrate emulation between WiFi, BLE, ZigBee, and LTE. Other works, such as XFi [19], XBee [12], and SymBee [24], explore transmitter- and receiver-transparent schemes, enabling parallel, high-speed CTC. PCTC [26] further improves performance by supporting parallel PHY-level transmissions. At higher layers, techniques such as Explicit Channel Coordination (ECC) [28] and X-MIMO [23] coordinate channel access and multi-user MIMO across heterogeneous networks, while PRComm [25], CrossZig [10], and CT-Jammer [20] address interference resilience and potential security vulnerabilities in multi-technology environments. However, these techniques do not apply to LoRa due to its unique CSS physical layer, which requires precise wideband linear chirps.

Recently, CTC has been explored specifically for LoRa networks. LoRaBee [22] enables cross-technology communication by encoding information from a LoRa transmitter into energy patterns detectable by co-located ZigBee receivers. WiLo [6] extends this concept to long-range scenarios, allowing Wi-Fi devices to communicate with LoRa receivers by shaping Wi-Fi signals to emulate LoRa-like chirps. Signal emulation has emerged as a particularly promising approach, offering high throughput and transparent communication across heterogeneous radios. BLE2LoRa [15] demonstrated that LoRa devices can interpret chirp-like signals emulated from BLE transmissions, enabling cross-technology ACKs. These studies show that LoRa's CSS modulation can be partially emulated by external radios to establish direct CTC.

However, existing CTC solutions based on signal emulation cannot be directly applied to LR-FHSS to LoRa communication due to several inherent physical-layer differences. LR-FHSS to LoRa CTC is fundamentally constrained by three factors: (i) the synthesized preamble, whether tone based or GMSK based, does not match the energy profile of native LoRa preambles, leading to unreliable CAD detection; (ii) FCC rules require at least a 25.4 kHz separation between LR-FHSS hops, which fragments any attempted chirp and disperses its energy across FFT bins; and (iii) the continuous phase evolution of LR-FHSS's GMSK modulation is inherently nonlinear, preventing alignment with the ideal linear chirp trajectory expected by a LoRa demodulator. Thus, conventional LoRa CTC techniques are not suitable for LR-FHSS to LoRa communication, and a novel technique specifically tailored for this scenario is needed. To the best of our knowledge, our CTC technique is the *first* to enable CTC from LR-FHSS to LoRa.

3 Background

This section summarizes the physical-layer properties of LR-FHSS and LoRa that are necessary to understand our LR-FHSS to LoRa CTC design. Although both modulations coexist within the same LoRaWAN radio platforms, they differ substantially in bandwidth usage, symbol structure, hopping behavior, and receiver processing. An overview of these differences aids in understanding the challenges addressed in this work.

3.1 LR-FHSS

LR-FHSS (Long Range–Frequency Hopping Spread Spectrum) is a narrowband, fast-hopping modulation introduced to support ultra-long-range uplinks and scalable massive-IoT deployments. An LR-FHSS transmission occupies a wide Operating Channel Width (OCW)—1.523 MHz in the US 902–928 MHz band—which is subdivided into thousands of 488 Hz occupied-bandwidth subcarriers. Regulatory requirements impose a minimum hop separation of 25.4 kHz, resulting in 52 grids of 60 usable subcarriers per OCW channel. A transmitting device selects one grid for the duration of a packet and transmits short GMSK-modulated fragments on subcarriers determined by a pseudo-random hopping sequence. This sequence is produced by hashing a per-packet seed and fragment index, ensuring rapid intra-packet frequency hopping and strong resilience to interference.

Each LR-FHSS packet begins with multiple GMSK-coded header replicas that allow the gateway to discover the hopping pattern. The number of header repetitions depends on the data rate mode: DR5 provides 162 bps with a coding rate of 1/3 and three header copies; DR6 provides 325 bps with a coding rate of 2/3 and two header copies. The payload is divided into 50 ms fragments, each transmitted at a different subcarrier location according to the hopping pattern. All fragments are sent consecutively without silent gaps. At the receiver, the gateway monitors all subcarriers in the OCW simultaneously, reconstructs the header, and reassembles the fragments into a full packet. Because the gateway processes all subcarriers in parallel, decoding capacity is governed largely by digital signal processing throughput. Summarily, LR-FHSS provides

exceptional link budget, resilience to narrowband interference, and scalability for high-density IoT networks, but it emits only discrete, narrowband GMSK fragments.

3.2 LoRa

LoRa's physical layer is anchored by a wideband *base up-chirp* that serves as the fundamental waveform for symbol generation and detection. LoRa employs Chirp Spread Spectrum (CSS), where each symbol is represented as an up-chirp whose instantaneous frequency increases linearly across the entire bandwidth. A symbol of spreading factor SF has duration $T_{\text{sym}} = 2^{SF}/BW$ and consists of a cyclic up-chirp spanning the full channel bandwidth. Information is encoded in the starting frequency of the chirp, effectively shifting the base up-chirp. At the receiver, the incoming signal is multiplied by a down-chirp, collapsing the frequency sweep into a single-tone sinusoid. The FFT over one symbol interval produces a strong peak at the corresponding symbol index, enabling robust demodulation at very low SNR.

A LoRa frame begins with several identical base up-chirps forming the preamble, followed by a Start-of-Frame Delimiter (SFD) composed of down-chirps that provide precise symbol boundary and timing alignment. CSS inherently offers high processing gain and strong robustness to multipath and interference. However, chirp generation requires continuous wideband waveform control and precise time–frequency structure. Signals that deviate from the ideal linear sweep, or that contain discontinuities in phase or frequency, spread energy across FFT bins and become undecodable.

Although LR-FHSS and LoRa are implemented in the same radio hardware, LR-FHSS's discrete, narrowband GMSK fragments are fundamentally different from the wideband chirp structure expected by LoRa receivers. Mandatory hop gaps clash with LoRa's dechirp-and-FFT pipeline. LR-FHSS cannot natively generate chirps, while LoRa downlinks cannot cover LR-FHSS uplink distances, motivating the exploration of transforming LR-FHSS transmissions into chirps interpretable by LoRa receivers.

4 Underlying Idea: Emulating Chirps via Sequential Hopping

The main idea of our CTC technique is the strategic manipulation of the LR-FHSS transmission framework to produce signals that, while not identical to LoRa chirps, possess a structural resemblance that can be exploited by a custom receiver. We aim to transform the chaotic, pseudo-random nature of LR-FHSS into the structured, deterministic frequency evolution of LoRa's CSS. Our approach is to repurpose the LR-FHSS PHY not as a data modulation engine, but as a flexible, multi-frequency waveform synthesizer.

As the first step to evaluate the feasibility of CTC communication between LR-FHSS and LoRa, we observe the impact of transmitting symbols on sequential subcarriers. Note that we relax the FCC requirement to demonstrate the challenges in CTC communication between LR-FHSS and LoRa. The proposed approach, described in Section 8, also handles the FCC requirement.

The transformation of the LR-FHSS signal to LoRa signal is achieved through two fundamental modifications at the gateway. First, we fundamentally alter the hopping behavior. Instead of utilizing the on-board pseudo-random number generator to select the next transmission subcarrier, we enforce a deterministic, **sequential hopping** pattern. To emulate a LoRa up-chirp, the gateway is programmed to transmit its signal fragments on a sequence of monotonically increasing subcarrier frequencies. This forces the time-frequency representation of the signal to change from a scattered plot into an ordered, ascending staircase pattern, which begins to approximate the linear sweep of a true chirp.

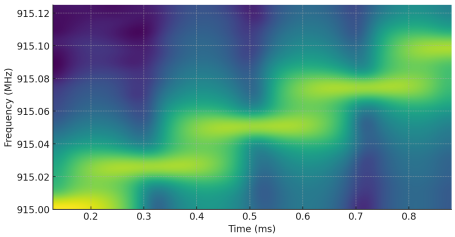


Figure 2: Spectrogram of a single symbol with only a few subcarriers carrying energy (e.g., five 25.4 kHz–spaced bursts). The sparse, stepped structure exposes the spectral discontinuity introduced by LR-FHSS hopping and foreshadows the need for reconstruction prior to LoRa dechirp and FFT.

Second, and critically, we address the modulation mismatch. The LoRa PHY uses frequency to encode symbols. The GMSK modulation used in LR-FHSS, being a form of phase modulation, would introduce complex phase variations that are alien to the LoRa demodulator and would obscure the underlying frequency information. Therefore, in our *primary* CTC design, we **bypass the GMSK modulator** entirely. For the duration of each fragment on a given subcarrier, the gateway transmits a simple, unmodulated carrier wave—a pure tone at that subcarrier's frequency. This simplification is vital; for each fragment, it ensures that the information is encoded onto the frequency. LoRa demodulation ultimately reduces each chirp to a tone whose *frequency bin* encodes the symbol; therefore, our pure-tone fragments preserve the frequency information needed for LoRa's dechirp+FFT. The signal synthesized through this process is termed **Pseudo-Chirp**. In Section 5, we show that the same architectural ideas can be extended to a GMSK-based Pseudo-Chirp when disabling GMSK is not possible.

Pseudo-Chirp is a piecewise-constant approximation of a continuous linear chirp. While this emulated signal preserves the essential information of a true chirp—its starting frequency and its rate of frequency change—its imperfect structure poses a significant challenge. To quantify this, we conducted a preliminary experiment using two USRP B200 SDRs in a controlled office environment. One USRP transmitted a single Pseudo-Chirp, and the other captured the raw I/Q samples. We then subjected this captured signal to the standard LoRa demodulation process (multiplication by a reference down-chirp followed by an FFT).

The signal's structure, visualized in the spectrogram in Figure 2, is the source of the problem. This inherent spectral discontinuity, as we will prove mathematically in the next section, is what causes the signal's energy to be smeared across a wide range of FFT bins, making it impossible for a standard LoRa receiver to decode the symbol. This empirical result provides a strong motivation for our core hypothesis: a successful CTC system requires an intelligent receiver capable of repairing this damaged signal structure before demodulation.

4.1 Mathematical Formulation of the Discontinuous Signal

To formally demonstrate *why* the Pseudo-Chirp is incompatible with a standard LoRa receiver, we must analyze its spectral properties after de-chirping.

An ideal LoRa up-chirp, encoding a symbol S with a corresponding starting frequency f_S, has a complex baseband signal $X_{\text{LoRa}}(t) = e^{j2\pi(f_S t + \frac{1}{2}kt^2)}$, where $k = BW/T_{sym}$ is the chirp rate.

During demodulation, this signal is multiplied by a reference down-chirp, $X_{\text{down}}(t) = e^{-j\pi kt^2}$. This "de-chirping" operation yields a pure sinusoid at the symbol frequency f_S:

$$Y_{\text{LoRa}}(t) = X_{\text{LoRa}}(t) \cdot X_{\text{down}}(t) = e^{j2\pi(f_S t + \frac{1}{2}kt^2)} \cdot e^{-j\pi kt^2} = e^{j2\pi f_S t}$$

The FFT of this signal, $\mathcal{F}\{Y_{\text{LoRa}}\}$, is an ideal delta function $\delta(f - f_S)$—a sharp peak at the correct frequency bin, allowing for trivial decoding.

Our CTC Pseudo-Chirp, $X_{\text{PC}}(t)$, is a collection of disjoint fragments. This can be modeled as the *ideal* LoRa chirp multiplied by a "windowing function," $W(t)$, which is equal to 1 during the transmission fragments and 0 in the spectral gaps between them.

$$X_{\text{PC}}(t) = X_{\text{LoRa}}(t) \cdot W(t)$$

Now, consider the de-chirping process for this signal:

$$Y_{\text{PC}}(t) = X_{\text{PC}}(t) \cdot X_{\text{down}}(t) = \big(X_{\text{LoRa}}(t) \cdot W(t)\big) \cdot X_{\text{down}}(t)$$

By rearranging terms, we see the true nature of the problem:

$$Y_{\text{PC}}(t) = \big(X_{\text{LoRa}}(t) \cdot X_{\text{down}}(t)\big) \cdot W(t) = Y_{\text{LoRa}}(t) \cdot W(t)$$

This means the de-chirped Pseudo-Chirp is the ideal pure sinusoid $e^{j2\pi f_S t}$ multiplied by the same discontinuous windowing function $W(t)$.

The final step at the receiver is the FFT. Using the convolution theorem, the spectrum of this signal is the convolution of the individual spectra:

$$\mathcal{F}\{Y_{\text{PC}}\} = \mathcal{F}\{Y_{\text{LoRa}}\} * \mathcal{F}\{W(t)\} = \delta(f - f_S) * \mathcal{W}(f) = \mathcal{W}(f - f_S)$$

where $\mathcal{W}(f)$ is the Fourier transform of the window function $W(t)$.

This result is definitive. The output of the LoRa receiver's FFT is **not** a sharp peak. Instead, it is the spectrum of the fragment windowing function, $\mathcal{W}(f)$, shifted to the correct symbol frequency f_S. Because $W(t)$ is a train of sharp, disjoint pulses, its spectrum $\mathcal{W}(f)$ is extremely broad, with energy distributed across many sidelobes. This precisely explains the consequence of the signal shown in Figure 2. Our analysis confirms that the inherent spectral discontinuity visualized in the spectrogram is not a simple phase error, but it is the fundamental challenge of the LR-FHSS to LoRa CTC. This analysis provides the core motivation for our Chirp Reconstruction Module (Section 9.2), which is designed to repair $W(t)$ *before* the final FFT is performed.

5 Alternative Formulation Using GMSK Modulation

While our primary CTC design bypasses GMSK modulation for simplicity and effectiveness, it is theoretically conceivable to construct a Pseudo-Chirp by manipulating the GMSK modulation itself. This alternative approach would adhere more closely to the LR-FHSS standard but presents significant practical challenges.

5.1 Underlying Principle: Instantaneous Frequency in GMSK

GMSK is a form of Continuous Phase Frequency Shift Keying (CPFSK) where the information is encoded in the phase changes of the carrier. The phase of a GMSK signal is given by:

$$\phi(t, \boldsymbol{\alpha}) = \phi_0 + \pi h \sum_{i=-\infty}^{\infty} \alpha_i q(t - iT_b)$$

where $\boldsymbol{\alpha} = \{\dots, \alpha_{-1}, \alpha_0, \alpha_1, \dots\}$ is the sequence of bipolar data symbols ($\alpha_i \in \{-1, 1\}$), $h = 0.5$ is the modulation index for GMSK, T_b is the bit period, and $q(t)$ is the phase response function, which is the integral of the Gaussian-filtered rectangular pulse.

The instantaneous frequency deviation from the carrier, $f_d(t)$, is the time derivative of the phase, scaled by $1/(2\pi)$:

$$f_d(t) = \frac{1}{2\pi}\frac{d\phi}{dt} = \frac{h}{2} \sum_{i=-\infty}^{\infty} \alpha_i g(t - iT_b)$$

Here, $g(t)$ is the Gaussian pulse shape. The total instantaneous frequency of the signal on a subcarrier f_c is $f_{\text{inst}}(t) = f_c + f_d(t)$. For a sequence of all +1s, the frequency deviation settles to $+h/(2T_b) = +1/(4T_b)$. For a sequence of all −1s, it settles to $-1/(4T_b)$.

5.2 Emulating a Chirp with GMSK Data

The core idea of this alternative approach is to control the *average* instantaneous frequency within each fragment period by carefully crafting the data bit sequence $\boldsymbol{\alpha}$ transmitted during that fragment. The coarse frequency steps of the chirp would still be provided by the sequential hopping of the subcarrier frequency $f_c(m)$. The fine-tuning to create a smoother sweep would be achieved by controlling the GMSK frequency deviation $f_d(t)$.

To emulate a linear chirp $f(t) = f_S + kt$, for each fragment m transmitted on subcarrier $f_c(m)$, we would need to select a data sequence $\boldsymbol{\alpha}_m$ such that the average instantaneous frequency matches the target:

$$\frac{1}{\Delta t}\int_{m\Delta t}^{(m+1)\Delta t} (f_c(m) + f_d(\tau, \boldsymbol{\alpha}_m))\, d\tau \approx f_S + k(m + 0.5)\Delta t$$

This implies that for fragments at the beginning of the Pseudo-Chirp, we need a data sequence with more negative ones to pull the average frequency down, and for fragments at the end, a sequence with more positive ones to pull it up.

5.3 Practical Challenges and Performance Trade-offs

While mathematically plausible, this GMSK-based approach introduces significant practical challenges compared to the pure-tone method. These challenges do not make the approach impossible, but they do result in a performance trade-off, namely a lower overall Packet Reception Rate (PRR) due to the imperfect nature of the emulated chirp. The primary challenges are as follows.

Limited Dynamic Range. The maximum frequency deviation available from GMSK is very small, only $\pm 1/(4T_b)$. For a typical LR-FHSS data rate, this deviation is on the order of a few tens of kHz. This is insufficient to smoothly fill the large spectral gaps (e.g., 25 kHz) between subcarriers. As a result, it is impossible to

perfectly emulate a smooth sweep, leading to residual phase noise and spectral artifacts that reduce the post-dechirp SNR.

Bit Sequence Generation Complexity. Determining the optimal bit sequence $\boldsymbol{\alpha}_m$ for each fragment to achieve a target average frequency is a complex optimization problem. This would require significant real-time computation at the gateway, which may not be feasible for all hardware platforms.

Receiver-Side Interference. A LoRa receiver is not designed to process a signal with rapid, data-induced frequency fluctuations around a central tone. The de-chirping operation interacts with these GMSK-induced variations, producing significant spectral spreading that is difficult to suppress fully, even with advanced signal processing at the receiver.

These challenges motivate the software-heavy mitigation techniques described next.

5.4 Solution for a GMSK-Based Pseudo-Chirp

(1) Average–deviation shaping with hop-step co-design. Because the GMSK deviation $f_d(t) = \frac{h}{2}\sum_i \alpha_i g(t - iT_b)$ is fundamentally bounded by $|f_d| \le \frac{h}{2T_b}$, we avoid the futile goal of instantaneously filling large inter–subcarrier gaps. Instead, each fragment of duration Δt is assigned a *target average* deviation $\bar{f}_m$ that gently steers the fragment's effective tone toward the ideal chirp trajectory. The bitstream $\boldsymbol{\alpha}_m \in \{-1, +1\}$ is produced by a first/second–order sigma–delta loop that integrates the error between the running estimate $\widehat{f_d}$ and $\bar{f}_m$, thereby concentrating quantization noise at higher temporal frequencies that the Gaussian pulse naturally attenuates. In parallel, we co-design the hop lattice and fragment granularity: increasing the number of hops M (shorter Δt) and, where permissible, refining the subcarrier raster reduce the per-fragment correction required from f_d. A light BT sweep (where BT is the Gaussian filter's bandwidth–time product in GMSK) and a single-tap pre-emphasis on $\boldsymbol{\alpha}_m$ preserve the low-frequency (mean) component of f_d while pushing residual jitter outside the LoRa post-dechirp passband.

(2) Real-time sigma–delta with run-length control. Exact per-fragment bit-pattern optimization against the Gaussian pulse is not viable under real-time constraints. We replace it with a causal two-line update that directly tracks the per-fragment mean:

$$e[n] = \bar{f}_m - \widehat{f_d}[n], \qquad \alpha[n] = \operatorname{sgn}\big(e[n] + \kappa_1 e[n-1] + \kappa_2 e[n-2]\big),$$

where $\widehat{f_d}[n]$ is the predicted deviation after filtering by $g(\cdot)$ and κ_1, κ_2 set the sigma–delta order. To prevent chatter and to stabilize the CPM memory, we enforce run-length bounds $L_{\min} \le \mathrm{runs}(\alpha) \le L_{\max}$ using a tiny codebook that maps the requested $\bar{f}_m$ to admissible patterns. This efficient, linear-time synthesis reliably matches the desired *average* deviation despite pulse shaping.

(3) CPM-aware dechirp to suppress intra-fragment flutter. LoRa's dechirp–FFT expects a near-tone, while CPM introduces smooth but data-dependent frequency flutter that spreads energy. We therefore insert a lightweight CPM front-end prior to the LoRa down-chirp. Using a Laurent decomposition, we reconstruct and subtract the dominant CPM component with parameters learned from the preamble. The down-chirp then multiplies by

$$X_{\text{down,CPM}}(t) = e^{-j\pi k t^2}\, e^{-j\widehat{\phi}_{\text{CPM}}(t)},$$

where $\widehat{\phi}_{\text{CPM}}(t)$ is synthesized from the known/decoded $\boldsymbol{\alpha}_m$ and the estimated Gaussian pulse. A small second-order PLL (or Kalman phase tracker) removes residual phase error across fragment boundaries. On the transmit side we maintain *phase-coherent* fragment transitions (no NCO resets, plus a 1–2 bit boundary guard chosen to align instantaneous phase slope), so the post-dechirp waveform concentrates cleanly in the correct FFT bin.

Offline precomputation and gateway lookup. To make the pipeline deployment-ready, we precompute a calibrated template bank and store it in the LR-FHSS gateway. Offline, we sweep the operating grid (subcarrier index, fragment index m, BT and pre-emphasis, coarse SNR tiers, and small CFO/temperature offsets) and, for each point, synthesize: (i) a sigma–delta + run-length–constrained $\boldsymbol{\alpha}_m^\star$ that realizes the target $\bar{f}_m$; (ii) the boundary-guard bits that guarantee phase-coherent transitions; and (iii) compact receiver side-information—the few Laurent coefficients and a parsimonious parametrization of $\widehat{\phi}_{\text{CPM}}(t)$—that enables CPM-aware dechirp. At runtime, the gateway selects the nearest template for the current hop/fragment and BT/pre-emphasis pair, applies at most an affine interpolation of $\bar{f}_m$ when the requested step falls between grid points, and emits the stored $\boldsymbol{\alpha}_m^\star$ with its guard.

Summary. Average–deviation shaping neutralizes the bounded $|f_d|$ constraint, sigma–delta with run-length control resolves real-time sequence design, and CPM-aware cancellation plus phase-coherent hopping addresses the receiver mismatch. With the offline template bank and on-gateway lookup, the system attains deterministic runtime and robust downstream LoRa decodability.

6 Key Technical Challenges in LR-FHSS to LoRa CTC

The endeavor of forcing two fundamentally disparate wireless technologies to communicate exposes a series of profound challenges. These are not mere implementation hurdles but incompatibilities at the physical layer that require non-trivial solutions. Overcoming them necessitates a comprehensive understanding of both the LoRa receiver's strict operational requirements and the constraints imposed by the LR-FHSS transmission framework.

6.1 Preamble Incompatibility and Receiver Activation

The first barrier to any communication is getting the receiver to listen. This challenge is particularly acute because a LoRa node is, by design, a parsimonious listener. It relies on a low-power Channel Activity Detection (CAD) mechanism which is not a generic energy detector but a highly specialized matched filter. This hardware filter is tuned to the specific time-frequency signature of a LoRa preamble—a sequence of 8 to 12 identical, unmodulated base up-chirps. The non-triviality of this challenge lies in the CAD's strictness; it requires a strong correlation against a local chirp replica. Any signal lacking this repetitive, sweeping energy profile is dismissed

as noise. Our synthesized Pseudo-Chirp, being a discrete approximation, presents a different energy signature. Therefore, a key research question is whether a sequence of these imperfect chirps can consistently accumulate enough correlation energy to pass the hardware CAD threshold, especially under varying SNR conditions. A failure to reliably trigger the CAD renders any subsequent communication impossible.

In practice, we ultimately resolve this incompatibility by shifting preamble detection to our firmware: instead of relying solely on the LoRa chip's built-in CAD matched filter, we merge many fragmented copies of a base up-chirp into a single coherent buffer, dechirp it, and look for a consistent FFT peak across multiple windows. This revised mechanism, detailed in Section 8.2 and Section 9.1, allows us to tolerate the fragmentation introduced by LR-FHSS while keeping the detection overhead extremely low.

6.2 Inherent Spectral Discontinuity

As empirically validated in Figure 2, the regulatory requirement of a frequency gap between consecutive LR-FHSS hops makes it impossible to generate a contiguous stepped-frequency signal. The resulting Pseudo-Chirp has periodic nulls in its spectrum. This is far more damaging than simple distortion. Let the spectrum of an ideal de-chirped signal be a narrow pulse $S(f)$. The presence of periodic gaps acts as a multiplication by a comb function, $C(f) = \sum_n \delta(f - n\Delta F)$, where ΔF is the frequency gap. The received spectrum is thus $S_{rx}(f) \approx S(f) \cdot C(f)$. In the time domain, this multiplication becomes a convolution with an impulse train, $s_{rx}(t) \approx s(t) * c(t)$, which smears the signal's energy across the symbol duration. When the final FFT is performed, this temporal smearing translates back to severe spectral leakage, scattering the energy across dozens of bins. This problem is non-trivial because it is an intrinsic structural flaw that cannot be removed by simple filtering; it requires actively reconstructing the missing spectral information.

6.3 Channel and Hardware Imperfections: CFO and CTO

The structural integrity of our Pseudo-Chirp is fragile and highly susceptible to real-world hardware imperfections. **Carrier Frequency Offset (CFO)**, arising from mismatched oscillator frequencies, is particularly pernicious. For a 125 kHz LoRa channel with Spreading Factor 7 (SF7), the spectrum is divided into 128 bins, each less than 1 kHz wide. A CFO of just 500 Hz—a common drift in COTS devices—can shift the signal energy into an adjacent FFT bin, guaranteeing a symbol error. The challenge is non-trivial because this offset must be estimated and corrected with a precision far greater than the bin width. **Coarse Timing Offset (CTO)** is equally damaging, as a timing error Δt in a chirp system is directly converted into a frequency error of $k \cdot \Delta t$ after de-chirping. Our system, which relies on the precise timing and frequency of discrete fragments, is doubly vulnerable. Any robust design must incorporate a high-precision offset correction mechanism as a mandatory, not optional, component.

6.4 Precise Time and Phase Synchronization

The LoRa de-chirping process is a coherent operation, making it exquisitely sensitive to phase. Our Pseudo-Chirp is assembled from disjoint fragments. A significant implementation challenge is ensuring phase continuity across these fragments at the transmitter, which requires direct, sample-accurate control over the SDR's Numerically-Controlled Oscillator (NCO) to perform phase-coherent frequency hopping, rather than allowing the oscillator phase to reset on each hop. At the receiver, the problem reappears. The reconstruction process must synthesize new signal samples while ensuring the phase of these segments is perfectly continuous with the received signal fragments. An abrupt phase jump acts as a high-frequency impulse, spreading energy across the entire spectrum during the FFT and effectively raising the noise floor. This degradation of the desired peak's SNR is non-trivial to avoid, demanding a meticulous phase-tracking and initialization process within the reconstruction algorithm.

LR-FHSS gateway (sender)	LoRa node (receiver)
Requires: (i) hop-sequence override, (ii) sample-accurate hop timing, (iii) phase-coherent frequency updates (NCO control) to avoid per-hop phase resets.	**Requires:** (i) access to complex I/Q samples, (ii) an insertion point *before* standard LoRa dechirp+FFT to run detection, offset correction, and CRM.
Not required: no RF front-end changes; no antenna/RF chain modification.	**Not required:** no RF front-end changes; LoRa demodulation remains unchanged after CRM output.

(a) Required changes.

Gateway (LR-FHSS TX) → FEC → preamble & payload pseudo-chirp construction → controlled hopping sequencer → RF → **Node (LoRa RX)** → I/Q buffer → preamble detect + coarse CFO → offset correction (CFO/CTO) → CRM reconstruction → **Standard LoRa**: dechirp → FFT → symbol decisions → FEC decode

(b) Pipeline.

Figure 3: (a) Sender and receiver requirements and (b) the end-to-end processing pipeline showing the boundary between our reconstruction and standard LoRa demodulation.

7 Technical Overview

We first make the sender/receiver requirements explicit, then summarize the end-to-end signal path and its interface boundary to the standard LoRa demodulation pipeline.

7.1 Required Changes and Assumptions

Our design assumes *PHY hooks* at both ends: baseband-level control on the LR-FHSS gateway and access to complex I/Q (plus a pre-demod insertion point) on the LoRa receiver. Importantly, no RF front-end modification is required. Figure 3 summarizes the sender/receiver requirements and the end-to-end pipeline, including the boundary to the standard LoRa dechirp+FFT.

7.2 COTS Feasibility and Limitations

Our prototype is SDR-based and we do not over-claim immediate COTS-to-COTS deployability without vendor firmware/baseband access. Commercial LR-FHSS gateways typically expose packet-level interfaces (insufficient for deterministic hop timing and phase continuity), and commodity LoRa transceivers typically expose decoded packets/metadata rather than raw I/Q and do not allow programmable preprocessing before dechirp+FFT.

Summary. With PHY hooks, the gateway synthesizes pseudo-chirps and the node reconstructs a clean chirp stream (CRM) that the unmodified LoRa demodulator can decode.

7.3 End-to-end Walkthrough

At the gateway, the **CTC Transmitter** architecture is designed to intercept a standard data packet and transform it into the structured Pseudo-Chirp waveform. Here, a user payload is first passed through a Forward Error Correction (FEC) encoder to build in resilience. A central control logic then processes the transmission, first engaging the Preamble Pattern Generator to synthesize the activation sequence, followed by the Symbol-to-Subcarrier Mapper for the data payload. The core of the design is the Sequential Hopping Sequencer and the PHY Control module, which together override the native LR-FHSS behavior to generate the phase-coherent, GMSK-bypassed pure tones that form the Pseudo-Chirp.

At the LoRa node, the received signal is processed by the **CTC Receiver**. After the signal is captured and digitized by the COTS RF front-end, our custom firmware continuously buffers the incoming I/Q samples and performs preamble detection in software, rather than relying solely on the chip's fixed CAD. Once a preamble is detected by our constructive merge–FFT detector and coarse CFO is estimated, the same firmware hands a synchronized window of I/Q samples to the Offset Correction Module, which refines CFO/CTO and aligns symbol boundaries. The corrected, synchronized samples are then fed to the critical Chirp Reconstruction Module (CRM), which transforms the imperfect Pseudo-Chirp into a high-fidelity, continuous linear chirp. Only after this restoration is the signal passed to the node's native LoRa demodulator. The resulting symbol decisions are then processed by a final FEC decoding stage to correct any residual errors and recover the original user data. This architecture effectively uses software to bridge the hardware incompatibility, allowing the unmodified LoRa demodulation hardware to function as intended.

8 CTC Transmitter

The transmitter functionality is implemented as a custom PHY layer on the gateway's SDR platform.

8.1 Forward Error Correction (FEC) Integration

To enhance resilience against noise and minor reconstruction errors, the transmitter first applies a systematic, block-based FEC code, such as a Reed-Solomon code, to the raw payload data. This adds redundant information that allows the receiver to detect and correct a certain number of symbol errors after demodulation, significantly improving the overall packet reception rate in challenging channel conditions.

8.2 Preamble Synthesis with Fragmented Pseudo-Chirps

To solve the receiver activation challenge under LR-FHSS fragmentation, the transmitter begins each packet by synthesizing a preamble that consists of p repeated *base up-chirps*, each of which is further split into multiple fragmented copies. Conceptually, each base up-chirp is the LoRa-style sweep we ultimately want the receiver to detect, but instead of transmitting it as a single contiguous chirp, we spread its energy over many short fragments.

Concretely, for a given base up-chirp, the Preamble Pattern Generator produces n *copies*, and each copy contains x non-overlapping fragments. Each fragment is realized as a short Pseudo-Chirp segment: a sequence of pure-tone hops over a local time-slice and frequency interval consistent with the global linear sweep. Across the n copies, these fragments collectively cover the full symbol duration of the base up-chirp in time, but, due to LR-FHSS hopping rules and FCC constraints (e.g., a required 25 kHz separation between consecutive hops even though the LR-FHSS raster is 488 Hz), each copy can only occupy a sparse subset of the available subcarriers. All fragments across all copies are chosen to be unique, so that the union of the $n \times x$ fragments progressively fills in different time–frequency tiles of the intended up-chirp.

For example, let's consider a practical configuration for BW = 125 kHz and SF = 7. In this mode, a single LoRa symbol (or base up-chirp) has a duration of $T_{sym} \approx 1.024$ ms. To ensure robust detection, we set the preamble length to $p = 10$ base up-chirps. The challenge is *how* we build one of those 10 symbols. We can't send a continuous chirp, so we build a "virtual" one by transmitting $n = 25$ "fragmented copies" **sequentially**. For *just one* of the $p = 10$ base chirps, the transmitter sends Copy 1 (with $x = 5$ fragments) over a 1.024 ms duration, followed by Copy 2 (with 5 *different* fragments) over the *next* 1.024 ms duration, and so on. This means the total transmission time for a single virtual base chirp is $n \times T_{sym}$, during which the receiver gathers all $n \times x = 125$ unique fragments. The full preamble consists of p such sequential virtual chirps.

At this stage of the communication, our goal is *coarse* detection rather than high-fidelity chirp reconstruction. The per-copy fragment count x is constrained by the LR-FHSS hopping and FCC spacing rule and is therefore the same for preamble and payload; instead, we deliberately use a *smaller* number of fragmented copies n for the preamble than for the payload. This means that the union of all $n \times x$ fragments coarsely covers the time–frequency support of the base up-chirp—sufficient for robust detection once the node merges the fragments—while payload symbols later use a larger number of copies to more densely fill the LoRa bandwidth for fine-grained reconstruction. To further increase robustness, the transmitter emits all preamble fragments at the *maximum* transmission power allowed by the FCC regulation (i.e., 30 dBm). Because this is downlink communication and the LR-FHSS gateway is not energy-constrained, we prioritize increasing PRR and reducing the node's energy consumption rather than saving energy at the gateway.

8.3 Payload Encoding and Generation

Following the preamble, the transmitter generates the data-carrying payload. This process transforms the FEC-encoded data into a sequence of higher-fidelity Pseudo-Chirps. A logical block, the **Symbol-to-Subcarrier Mapper**, takes an input symbol and maps it to an initial subcarrier frequency f_{start}. A **Sequential Hopping Sequencer** then generates the deterministic list of subsequent subcarrier frequencies for the chirp.

While the preamble uses a minimal number of fragmented copies (n_{preamble}) to enable rapid detection (as described in Sec 8.2), the payload symbols are synthesized using a much larger number of fragmented copies, n_{payload}, where $n_{\text{payload}} > n_{\text{preamble}}$. This higher effective sampling density in the time–frequency plane provides the Chirp Reconstruction Module (CRM) with a denser set of data

points, allowing it to perform a more accurate regression and reconstruct a near-ideal chirp.

Finally, the **PHY Control and Tone Generation** block executes this sequence. It iterates through the subcarriers, disabling the GMSK modulator and instructing the SDR to generate a pure carrier tone. Critically, to address the challenge of phase continuity at the transmitter, this module is designed to directly manipulate the SDR's NCO, ensuring phase-coherent frequency transitions between fragments rather than allowing the oscillator phase to reset, a non-trivial implementation detail for many commodity SDRs.

9 CTC Receiver

The receiver's intelligence is encapsulated in its firmware, which adds three critical software modules to the standard LoRa processing chain.

9.1 Preamble-Based Detection and Offset Correction

The revised preamble format requires a lightweight detection strategy that is much faster than the full-symbol reconstruction used for the payload. **Therefore, this module explicitly bypasses the CRM** and, instead of matching against contiguous up-chirps, our receiver constructs them *virtually* by aggregating fragments.

This process operates over a longer time scale than a single symbol. To detect a *single* virtual base up-chirp, the receiver must buffer n consecutive T_{sym} windows, corresponding to the total duration of the virtual chirp, $T_{virtual} = n \times T_{sym}$.

For each candidate virtual chirp window of duration $T_{virtual}$, the receiver gathers the n fragmented copies (totaling $n \times x$ fragments) from its buffer. Let $r_\ell[t]$ denote the captured complex baseband samples for the ℓ-th copy ($\ell = 1, \ldots, n$), with fragment start indices $t_{\ell,k}$ and lengths $L_{\ell,k}$. The receiver creates a merged buffer $u[t]$ by adding these fragments *constructively*.

$$u[t] = \sum_{\ell=1}^{n} \sum_{k=1}^{x} \mathbb{I}_{t \in [t_{\ell,k},\, t_{\ell,k}+L_{\ell,k})}\; r_\ell[t]\, e^{-j\psi_{\ell,k}[t]}.$$

where $\psi_{\ell,k}[t]$ is a low-complexity phase "stitching" term that roughly aligns the fragment's phase with the ideal up-chirp. Two practical choices are:

- *Midpoint linearization:* approximate the local phase with a linear model using the ideal instantaneous frequency at the fragment midpoint and continue phase from the end of the previous fragment.
- *Down-chirp pre-compensation:* approximate $\psi_{\ell,k}[t] \approx \pi k(t - t_{\ell,k})^2$ so that subsequent multiplication by the reference down-chirp cancels most of the quadratic term.

Because preamble detection is coarse, this phase model does not need to be perfect; its role is simply to make overlapping fragments add constructively rather than destructively.

The merged buffer $u[t]$, which contains fragments spanning $n \times T_{sym}$, is then "folded" back into a single T_{sym} window. This merged signal is then multiplied by the reference down-chirp, $X_{\text{down}}(t) = e^{-j\pi k t^2}$, to produce $v[t] = u[t]X_{\text{down}}(t)$, and a single FFT is computed over the T_{sym} duration. The receiver records the dominant FFT bin index $i^\star$ and its magnitude. This process is repeated for all $p = 10$ consecutive virtual chirp windows.

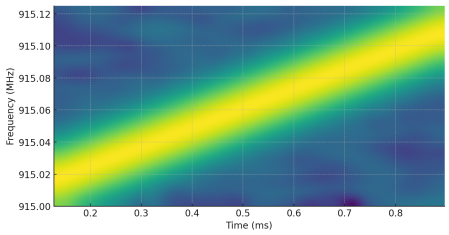
We then apply a *configurable threshold-based decision rule*. We define a detection threshold $p_{thresh} \le p$ (e.g., $p_{thresh} = 3$). If at least p_{thresh} of the $p = 10$ windows report the **exact same dominant FFT bin**, we declare that a preamble is present, and the corresponding time interval becomes our coarse timing reference. This rule is more flexible than a simple majority and is robust to occasional fragment loss, which may corrupt a subset of windows but is unlikely to force consistent peaks in an incorrect bin.

The same voting structure gives us a low-cost CFO estimate. Let $f_{\text{bin}} = BW/2^{SF}$ be the FFT bin spacing, and let i_0 be the expected de-chirped bin index for a zero-offset base up-chirp. The CFO shifts all peaks by approximately the same offset Δi, so we estimate CFO as

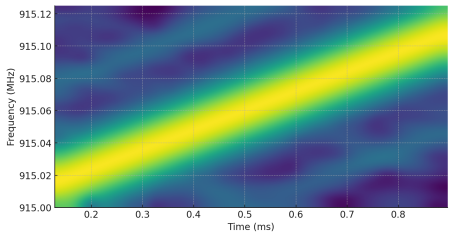
$$\widehat{\text{CFO}} = \text{mode}\big((i_w^\star - i_0) \cdot f_{\text{bin}}\big), \quad w = 1, \ldots, p,$$

(calculated only from the p_{thresh} or more windows that agreed on the peak), optionally refined by a small parabolic interpolation around each $i_w^\star$ for sub-bin accuracy. Because this estimator reuses the FFTs we already computed for detection, it adds almost no overhead. If, in extremely adverse conditions, the voting fails to produce a stable bin index, the receiver falls back to our earlier CFO estimator based on phase rotation between consecutive merged preamble windows.

Finally, this preamble detection event provides the coarse symbol boundary. Using the same merged buffers across the p up-chirps, the Offset Correction Module refines the timing alignment and applies the estimated CFO as a digital counter-rotation to all subsequent I/Q samples. At this point, the system has a consistent frequency base and symbol clock, and the CRM can operate on clean, synchronized payload symbols.



(a) Ideal LoRa chirp



(b) After processing by the CRM

Figure 4: Spectrogram comparison of (a) an ideal, continuous LoRa up-chirp and (b) our high-fidelity reconstructed chirp after processing by the CRM.

It is important to note that this coarse detection strategy is well-suited for standard LoRaWAN use cases (e.g., Class A), where a

node does not need to continuously monitor the channel but rather opens a brief receive (Rx) window after a transmission for downlink messages. For such scenarios, our lightweight preamble detection is perfectly adequate. However, for a node or gateway with no power constraints, this design can be modified for continuous listening. This would involve a sliding window, advancing by one virtual-chirp duration ($T_{virtual} = n \times T_{sym}$) at a time, and running the detection algorithm on the last p windows. This allows the node to always detect any incoming packet with a coarse estimation. If needed, this high-power mode could also be modified to utilize the CRM for a fine-grained preamble estimation, though this would significantly increase the computational load.

9.2 The Chirp Reconstruction Module (CRM)

After offset correction, the CRM receives a clean, synchronized stream of I/Q samples for each payload symbol. Its purpose is to reconstruct a continuous, high-fidelity chirp from the fragmented Pseudo-Chirp structure imposed by the LR-FHSS transmitter. This module is at the core of our CTC technique: it is responsible for transforming the discontinuous, staircase-like fragment sequence into a waveform that resembles a true CSS chirp closely enough for the native LoRa demodulator to process without modification.

Fragment identification and the case for lightweight analysis. The first step in reconstruction is to identify the set of fragment frequencies present within the symbol. We deliberately adopt a light, hardware-friendly approach based on short-time spectral analysis. Specifically, the CRM computes a sliding-window spectrogram over the symbol's I/Q samples using small FFTs. In each time slice, the module identifies the frequency bin with the dominant energy, producing a sequence of time–frequency points that locate the observed fragments.

This approach is a critical design choice. While high-resolution subspace methods (e.g., MUSIC or ESPRIT) could, in principle, estimate frequencies with sub-bin precision, they demand covariance matrices and eigenvalue decompositions. In the LoRa node environment—with its tight RAM and compute budgets—such $O(L^3)$ operations are prohibitively expensive in both energy and time. In contrast, our $O(L \log L)$ FFT-based peak tracking is computationally stable, requires negligible memory, and maintains deterministic timing. Because the Pseudo-Chirp is strong at the active frequencies and silent elsewhere, this lightweight method achieves excellent detection performance while remaining practical for resource-constrained LoRa hardware.

Robust chirp model estimation. Once the CRM has extracted the set of fragment points from the spectrogram, it estimates the underlying chirp parameters. The collection of detected points typically follows an approximately linear trajectory in the time–frequency plane. To account for occasional misdetections or noise, the CRM applies a robust regression procedure to fit a straight line:

$$\hat{f}(t) = \hat{f}_{start} + \hat{k}'t,$$

thereby estimating both the starting frequency and the effective chirp rate. This line fitting is extremely resilient, since even a sparse subset of correctly detected fragments suffices to recover the chirp parameters with high accuracy. The regression itself is computationally trivial compared to the subsequent signal synthesis, making it well suited to embedded environments.

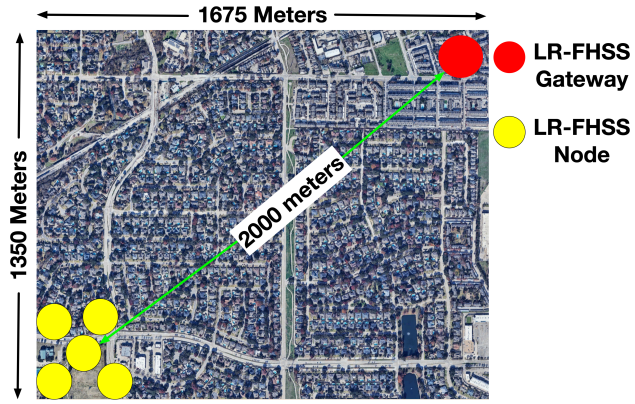


Figure 5: Outdoor Experimental setup.

Phase-coherent signal synthesis. With $(\hat{f}_{start}, \hat{k}')$ determined, the CRM constructs a synthetic chirp that fills in all fragment gaps. For each missing sample interval, the module analytically generates the ideal chirp phase:

$$\hat{\phi}(t) = 2\pi\left(\hat{f}_{start}t + \frac{1}{2}\hat{k}'t^2\right),$$

and produces the corresponding complex baseband samples $\hat{s}(t) = e^{j\hat{\phi}(t)}$. To ensure the full symbol remains phase-coherent, each synthesized segment is initialized using the final phase of the last real fragment at its boundary. This prevents phase discontinuities that would otherwise introduce broadband spectral leakage during the LoRa FFT.

The final reconstructed waveform is thus a hybrid signal: original received fragments preserved where available, and synthetic samples inserted seamlessly where gaps existed. This continuous, phase-consistent chirp is then passed to the standard LoRa demodulator.

Effectiveness in practice. Applying our reconstruction pipeline to the captured Pseudo-Chirp in the preliminary experiment yields a reconstructed waveform that closely follows the ideal LoRa up-chirp (Fig. 4). Despite the heavy fragmentation in the original transmission, the CRM restores a continuous chirp whose time–frequency trajectory is almost indistinguishable from that of a clean LoRa symbol. From the FFT's perspective, the reconstructed chirp behaves equivalently to an ideal one. This result confirms that our lightweight, embedded-friendly CRM—designed without computationally expensive estimators—achieves the required reconstruction fidelity while remaining practical for resource-constrained LoRa nodes.

9.3 Symbol Demodulation and FEC Decoding

The native LoRa hardware, now fed a high-quality chirp by the CRM, performs its standard de-chirp and FFT operation, producing an energy peak in the correct symbol bin. The sequence of detected symbols is then passed to the software-based **FEC Decoder**. This module uses the redundancy added at the transmitter to identify and

correct any errors that may have slipped through the reconstruction and demodulation process, finally outputting the original, error-free user payload.

10 Experiments

This section evaluates our LRFHSS to LoRa CTC design in a real outdoor deployment. We first describe the setup and metrics. We then present a number of experiments to measure the effect of bandwidth, spreading factor, transmit power, distance, and payload size on reliability and cost of our CTC technique. Next, we quantify end-to-end network performance (PRR, goodput, and energy per packet) by incorporating ACK in LR-FHSS network with our CTC technique and compare with other reliability policies. Finally, we measure how the system scales as the network size grows. Throughout, "ours" denotes the proposed LR-FHSS to LoRa CTC technique, while "LoRa" and "LR-FHSS" lines show native baselines where applicable.

10.1 Setup

We deploy one LR-FHSS gateway, and five LR-FHSS nodes with LoRa reception in a suburban metropolitan area as shown in Figure 5. USRP B200s [2] are used for all the gateways and nodes. We impelemt our technique on the USRPs using GNU Radio [3]. In default we use center frequency of 902–915 MHz, LoRa bandwidth (BW) of 125 kHz, and the default transmit powers of 0 dBm for nodes and 30 dBm for the gateway. Unless stated otherwise, each configuration point is repeated 10 times, and each repetition contains 100 packets (total 1000 packets per point). In each repetition we transmit 100 packets of size 10 Bytes with a 10 seconds gap between packet transmissions. We report mean values computed across repetitions.

Metrics. We report three metrics that together capture reliability, throughput, and energy cost:

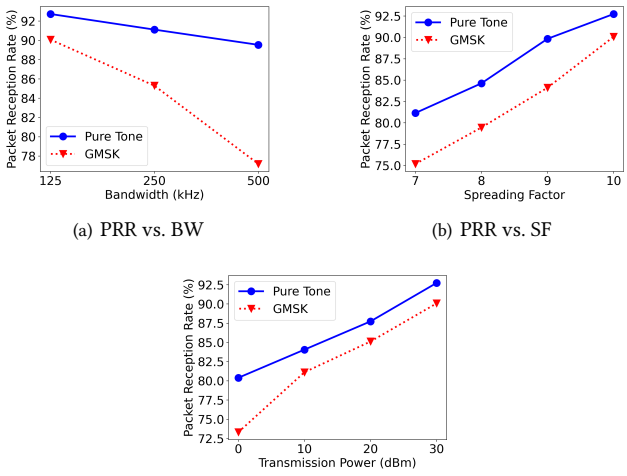
- **Packet Reception Rate (PRR)**: PRR = $\frac{N_{\text{rcv}}}{N_{\text{tx}}}$.
- **Goodput (bits/s)**: payload bits delivered per second, computed from the received payload size, ToA, and redundant packets.
- **Energy Per Packet (EPP, mJ)**: EPP = $\frac{E_{\text{Tx}}+E_{\text{ReTx}}+E_{\text{Rx}}}{N_{\text{rcv}}}$.

where, N_{tx} – total transmitted packets, N_{rcv} – total received packets, E_{Tx} – energy for transmissions, E_{ReTx} – energy for retransmissions, E_{Rx} – energy spent receiving (ACKs, beacons, etc.). Higher PRR and goodput, and lower EPP means better performance.

10.2 Evaluation of Our Technique

We first stress individual PHY/MAC parameters to understand how LRFHSS to LoRa CTC behaves.

10.2.1 Bandwidth (BW). Figure 6(a) shows PRR as LoRa BW is varied from 125 kHz to 500 kHz. As expected, narrower BW improves SNR per FFT bin and yields the highest PRR at 125 kHz. With wider BW, LoRa's symbol time shortens and the symbol energy spreads over more FFT bins; the LR-FHSS tone's per-hop dwell also contributes less energy per bin. These effects tighten the timing and frequency tolerances at the receiver, making our CRM more difficult.



(a) PRR vs. BW (b) PRR vs. SF

(c) PRR vs. Tx power

Figure 6: Sensitivity to bandwidth, spreading factor, and transmit power.

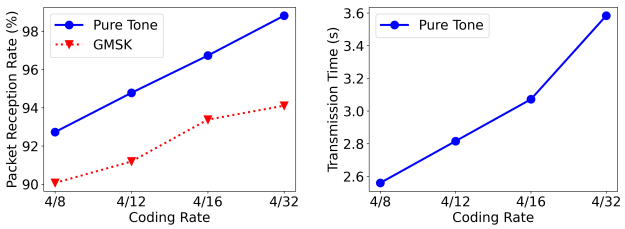
Even so, PRR degrades only mildly and our CTC technique maintains ∼89–92% across the sweep, indicating robust demodulation even as the bandwidth increases.

10.2.2 Spreading Factor (SF). Figure 6(b) varies LoRa SF from 7 to 10. Our CTC technique benefits from the longer symbol duration and larger processing gain at higher SF; PRR rises monotonically (about 81% to 93%). While using GMSK (dotted red), the gap narrows with SF because longer LoRa symbols give more integration time to accumulate cross-technology energy. Still, the pure-tone guided design is consistently superior, matching the intuition from the technical section that aligning LRFHSS hops with LoRa FFT windows maximizes detectability.

10.2.3 Transmit Power. In Figure 6(c), PRR improves for both waveforms as node power increases from 0 to 30 dBm. Our CTC maintains a margin of 3–5% over the proxy at every operating point due to better tone-to-bin alignment and the **effectiveness of our Chirp Reconstruction Module (CRM) in restoring a high-SNR signal (Sec 9.2)**. These results validate the SNR-scaling predicted by our model.

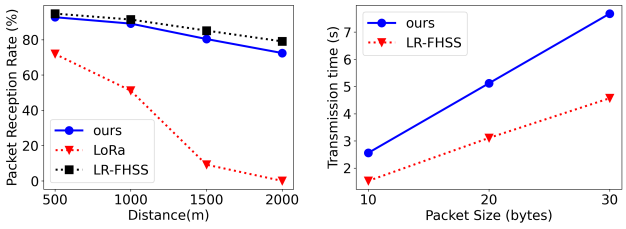
10.2.4 Receiver Overhead (Proxy Estimate). We estimate the runtime and memory footprint of the synchronization+CRM pipeline using a Python/Numpy proxy (scripts/receiver_overhead_estimate.py). On a desktop CPU, the proxy processes one SF7/BW125 symbol in **0.63 ms** (median over 200 runs; p90 **0.70 ms**) and uses about **70 KB** of working memory. Since we cannot directly measure energy on embedded LoRa hardware in time, we provide a conservative feasibility bound by tying energy to compute: $E_{\text{proc}} \approx P_{\text{active}} \cdot t_{\text{proc}}$ with t_{proc} from above and P_{active} = SENSYS mW for a LoRa-class MCU.

10.2.5 FEC Redundancy: Reliability–Delay Trade-off. Figures 7(a) and 7(b) plot the impact of our **external Forward Error Correction (FEC) rate** on reliability and airtime. We sweep the **block code rate (e.g., a Reed-Solomon** $R_c = K/N$**)** from 4/8 (least redundancy) to 4/32 (most redundancy), keeping other parameters



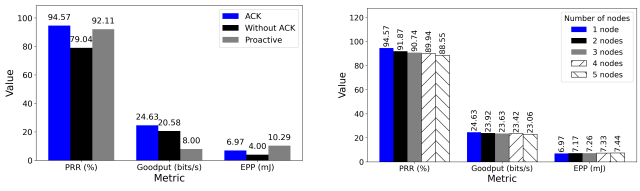
(a) PRR vs. FEC rate (b) Transmission time vs. FEC rate

Figure 7: FEC rate sweep. Higher redundancy (lower R_c) improves PRR at the cost of longer airtime.



(a) PRR vs. distance (b) Tx time vs. payload size

Figure 8: Path-loss and payload effects.



(a) Network metrics (b) Scalability

Figure 9: Network-level performance metrics (left) and scalability as the number of nodes increases (right).

fixed. **This is the systematic block code described in Sec 7, not LoRa's native convolutional code.**

Observations. PRR increases monotonically with added redundancy: for our pure-tone CTC path PRR rises from ~92.7% at $R_c = 4/8$ to ~98.8% at $R_c = 4/32$, and the GMSK ablation follows the same trend but stays below our curve (Fig. 7(a)). The cost is a linear increase in transmission time (Fig. 7(b)): from ~2.56 s at $R_c = 4/8$ to ~3.58 s at $R_c = 4/32$ for the tested payload and PHY settings. This behavior matches the model—lower code rate adds parity symbols, which raises processing gain and decoding margin, while airtime grows roughly in proportion to $1/R_c$.

How this compares to other knobs (BW, SF, Tx). Increasing SF (Fig. 6(b)) also boosts PRR but often stretches airtime more aggressively than moderate FEC rate changes because $T_{\text{sym}} \propto 2^{\text{SF}}/BW$. Narrowing BW (Fig. 6(a)) yields a small PRR gain with negligible control overhead but reduces instantaneous capacity. Raising Tx power (Fig. 6(c)) improves PRR without extra delay but increases energy. In contrast, our external FEC offers a reliability–delay trade-off with predictable, linear airtime growth and some extra power.

10.2.6 Varying Distance. Figure 8(a) compares PRR versus link distance for three systems (not specifically uplink or downlink; we simply consider three transmitter–receiver pairs): i) a native LoRa transmitter and LoRa receiver (denoted LoRa), ii) a native LR-FHSS transmitter and LR-FHSS receiver (denoted LR-FHSS), and iii) an LR-FHSS transmitter and a LoRa receiver equipped with our CTC technique (denoted ours). LoRa degrades sharply beyond 1 km, whereas LR-FHSS remains resilient due to frequency hopping and its ultra-narrow bandwidth. Our CTC technique (blue) closely tracks LR-FHSS across the entire range—typically within a few percentage points—while significantly outperforming native LoRa. This aligns with our design intuition: even when individual hops experience fading, **the Chirp Reconstruction Module (CRM) can perform a robust regression using the remaining fragments (Sec. 9.2)** to synthesize a high-quality chirp that the LoRa demodulator can reliably decode.
Takeaway. These results show that our approach *preserves the long range robustness of LR-FHSS* while *enabling standard LoRa reception*, enabling LR-FHSS-to-LoRa reception at distances where LoRa alone fails.

10.2.7 Payload Size and Transmission Time. Figure 8(b) plots transmission time versus payload size (varying from 10 to 30 Bytes) and compares our CTC technique (LR-FHSS to LoRa) with native LR-FHSS to LR-FHSS. Both configurations use a fixed 8 Bytes preamble and a compact LR-FHSS schedule. As a result, total transmission time increases with payload but remains predictable. Our CTC incurs a higher transmission time because it carries duplicated symbol copies, yet the growth is *linear*, not exponential.

10.3 End-to-end network behavior

We move beyond link tests to evaluate end-to-end behavior in an LR-FHSS network under three reliability policies: (i) *ACK* (downlink acknowledgments with limited retries), (ii) *Without ACK* (fire-and-forget), and (iii) *Proactive* (duplicate uplinks without feedback). Uplink packets are sent by LR-FHSS nodes; the gateway confirms reception by transmitting ACKs using our CTC transmitter so that the nodes' LoRa receivers can hear the downlink.

Deployment and policies. Nodes transmit a 10 B uplink every 10 s. After each uplink, the node either (i) opens a short Rx window and retries on ACK failure (ACK; two retries), (ii) never listens or retries (Without ACK), or (iii) sends two scheduled duplicates of the same uplink without waiting for downlink (Proactive). The gateway de-duplicates packets when computing goodput.

PRR, goodput, and EPP trade-offs. Figure 9(a) shows that *ACK* achieves the best overall performance, with PRR 94.57% and the highest goodput, at moderate EPP driven by occasional retransmissions and short receive windows. *Without ACK* reduces Rx cost but PRR drops to 79.04%, which directly reduces goodput. *Proactive* attains high PRR via duplication, but goodput decreases and EPP increases because duplicates consume airtime and energy even when the first copy succeeds.

Takeaway. When downlink capacity permits, ACK enabled by our CTC technique provides the most reliable and efficient operating point.

Link-budget context. The long-range gap between LoRa and LR-FHSS is consistent with effective-noise-bandwidth differences: for the same received power, the post-demod SNR scales with $10 \log_{10} B$ (integrated noise), while LR-FHSS concentrates energy into ultra-narrow hops and LoRa spreads energy over BW and relies on CSS processing gain. Our CTC design targets LR-FHSS's link margin by transmitting energy as structured hop fragments and reconstructing a LoRa-decodable chirp at the receiver.

10.4 Scalability

We extend the setup from Section 10.3 and evaluate scaling under the *ACK* policy (since Without-ACK PRR is low). Figure 9(b) reports PRR, goodput, and EPP as the number of nodes increases from one to five. PRR decreases modestly (94.6% → 88.6%), goodput follows a gentle decline (24.63 → 23.06 bits/s), and EPP increases slightly (6.97 → 7.44 mJ). The drop with five nodes is primarily driven by higher offered load: more uplinks increase collision probability and create occasional ACK scheduling pressure (missed/delayed ACKs), which triggers extra retransmissions and slightly raises EPP.

11 Conclusion

This paper addressed the downlink asymmetry in LR-FHSS deployments, where commercial LR-FHSS nodes are transmit-only and, although they contain LoRa receivers, LoRa downlinks fail at the extended ranges where LR-FHSS excels. As a result, gateways cannot provide reliability feedback or reconfigure remote LR-FHSS nodes, constraining the practicality of long-range or satellite-assisted IoT systems. We presented a CTC technique that enables long-range *LR-FHSS to LoRa* communication without RF front-end changes, assuming PHY-hook access at both the LR-FHSS gateway and the LoRa receiver.. We implemented our proposed CTC technique using USRP B200 with GNU Radio and evaluated it across diverse outdoor scenarios. Across scenarios, our design improved long-range reliability by up to 79% over LoRa and increased end-to-end LR-FHSS PRR by about 15% by enabling ACK. Our approach repurposes the LR-FHSS PHY to synthesize LoRa-compatible pseudo-chirps and uses a lightweight Chirp Reconstruction Module (CRM) so commodity LoRa demodulation can decode them.

Acknowledgments

The work was supported by the US National Science Foundation through grants CNS-2601685 and CNS-2602744, and by the US Office of Naval Research through grant N00014-23-1-2151.

References

[1] [n.d.]. https://blog.semtech.com/lorawan-protocol-expands-network-capacity-with-new-long-range-frequency-hopping-spread-spectrum-technology.
[2] [n.d.]. Ettus Research. https://www.ettus.com/product/.
[3] [n.d.]. GNU Radio. http://gnuradio.org.
[4] Zhenlin An, Qiongzheng Lin, and Lei Yang. 2018. Cross-Frequency Communication: Near-Field Identification of UHF RFIDs with WiFi!. In *MobiCom* (New Delhi, India) *(MobiCom '18)*. Association for Computing Machinery, New York, NY, USA, 623–638. https://doi.org/10.1145/3241539.3241569
[5] Hsun-Wei Cho and Kang G. Shin. 2021. BlueFi: Bluetooth over WiFi. In *SIGCOMM* (Virtual Event, USA) *(SIGCOMM '21)*. Association for Computing Machinery, New York, NY, USA, 475–487. https://doi.org/10.1145/3452296.3472920
[6] Demin Gao, Haoyu Wang, Shuai Wang, Weizheng Wang, Zhimeng Yin, Shahid Mumtaza, Xingwang Li, Valerio Frascolla, and Arumugam Nallanathan. 2024. WiLo: Long-Range Cross-Technology Communication from Wi-Fi to LoRa. *IEEE Trans. Commun.* (09 2024). https://doi.org/10.1109/TCOMM.2024.3461574
[7] Md Ashikul Haque and Abusayeed Saifullah. 2024. Handling jamming attacks in a LoRa network. In *IoTDI*. IEEE, 146–157.
[8] Md Ashikul Haque and Abusayeed Saifullah. 2025. Mitigating Jamming Attacks in LoRa Networks: A Defense Strategy against LoRa-Based Jammers. In *MobiHoc*. 51–60.
[9] Md Ashikul Haque, Abusayeed Saifullah, and Haibo Zhang. 2025. Deep Reinforcement Learning Based Coexistence Management in LPWAN. *INFOCOM* (2025).
[10] Anwar Hithnawi, Su Li, Hossein Shafagh, James Gross, and Simon Duquennoy. 2016. CrossZig: Combating Cross-Technology Interference in Low-Power Wireless Networks. In *IPSN*. 1–12. https://doi.org/10.1109/IPSN.2016.7460663
[11] Aakriti Jain, Md Ashikul Haque, Abusayeed Saifullah, and Haibo Zhang. 2024. Burst-MAC: A MAC Protocol for Handling Burst Traffic in LoRa Network. In *RTSS*. 148–160. https://doi.org/10.1109/RTSS62706.2024.00022
[12] Wenchao Jiang, Song Min Kim, Zhijun Li, and Tian He. 2018. Achieving Receiver-Side Cross-Technology Communication with Cross-Decoding. In *MobiCom* (New Delhi, India) *(MobiCom '18)*. Association for Computing Machinery, New York, NY, USA, 639–652. https://doi.org/10.1145/3241539.3241547
[13] Wenchao Jiang, Zhimeng Yin, Ruofeng Liu, Zhijun Li, Song Min Kim, and Tian He. 2017. BlueBee: A 10,000x Faster Cross-Technology Communication via PHY Emulation. In *SenSys* (Delft, Netherlands) *(SenSys '17)*. Association for Computing Machinery, New York, NY, USA, Article 3, 13 pages. https://doi.org/10.1145/3131672.3131678
[14] Song Min Kim and Tian He. 2015. FreeBee: Cross-Technology Communication via Free Side-Channel. In *MobiCom* (Paris, France) *(MobiCom '15)*. Association for Computing Machinery, New York, NY, USA, 317–330. https://doi.org/10.1145/2789168.2790098
[15] Zhijun Li and Yongrui Chen. 2020. BLE2LoRa: Cross-Technology Communication from Bluetooth to LoRa via Chirp Emulation. In *SECON*. 1–9. https://doi.org/10.1109/SECON48991.2020.9158446
[16] Zhijun Li and Tian He. 2017. WEBee: Physical-Layer Cross-Technology Communication via Emulation. In *MobiCom* (Snowbird, Utah, USA) *(MobiCom '17)*. Association for Computing Machinery, New York, NY, USA, 2–14. https://doi.org/10.1145/3117811.3117816
[17] Zhijun Li and Tian He. 2018. LongBee: Enabling Long-Range Cross-Technology Communication. In *INFOCOM*. 162–170. https://doi.org/10.1109/INFOCOM.2018.8485938
[18] Ruofeng Liu, Zhimeng Yin, Wenchao Jiang, and Tian He. 2019. LTE2B: Time-Domain Cross-Technology Emulation under LTE Constraints. In *SenSys* (New York, New York) *(SenSys '19)*. Association for Computing Machinery, New York, NY, USA, 179–191. https://doi.org/10.1145/3356250.3360022
[19] Ruofeng Liu, Zhimeng Yin, Wenchao Jiang, and Tian He. 2020. XFi: Cross-technology IoT Data Collection via Commodity WiFi. In *ICNP*. 1–11. https://doi.org/10.1109/ICNP49622.2020.9259363
[20] Ziwei Liu, Linzhong Xu, Feng Lin, Zhangsen Wang, and Kui Ren. 2022. CTJammer: A Cross-Technology Reactive Jammer towards Unlicensed LTE. In *IoTDI*. 95–106. https://doi.org/10.1109/IoTDI54339.2022.00013
[21] Tuofu Lu. 2020. LoRaWAN® Protocol Expands Network Capacity with New Long Range–Frequency Hopping Spread Spectrum Technology. *SEMTECH, URL: https://blog. semtech. com/lorawan-protocol-expands-network-capacity-withnew-long-range-frequency-hopping-spread-spectrum-technology.[Online]. Accessed* 7 (2020), 2024.
[22] Junyang Shi, Di Mu, and Mo Sha. 2019. LoRaBee: Cross-Technology Communication from LoRa to ZigBee via Payload Encoding. In *ICNP*. 1–11. https://doi.org/10.1109/ICNP.2019.8888145
[23] Shuai Wang, Woojae Jeong, Jinhwan Jung, and Song Min Kim. 2020. X-MIMO: Cross-Technology Multi-User MIMO. In *SenSys* (Virtual Event, Japan) *(SenSys '20)*. Association for Computing Machinery, New York, NY, USA, 218–231. https://doi.org/10.1145/3384419.3430723
[24] Shuai Wang, Song Min Kim, and Tian He. 2018. Symbol-Level Cross-Technology Communication via Payload Encoding. In *ICDCS*. 500–510. https://doi.org/10.1109/ICDCS.2018.00056
[25] Wei Wang, Dingsheng He, Wan Jia, Xiaojiang Chen, Tao Gu, Haiyan Liu, Xiaoyang Sun, Guannan Chen, and Fuping Wu. 2021. PRComm: Anti-Interference Cross-Technology Communication Based on Pseudo-Random Sequence. In *IPSN* (Nashville, TN, USA) *(IPSN '21)*. Association for Computing Machinery, New York, NY, USA, 163–175. https://doi.org/10.1145/3412382.3458264
[26] Jialiang Yan, Siyao Cheng, Zhijun Li, and Jie Liu. 2022. PCTC: Parallel Cross Technology Communication in Heterogeneous wireless systems. In *IPSN*. 67–78. https://doi.org/10.1109/IPSN54338.2022.00013
[27] Zhimeng Yin, Wenchao Jiang, Song Min Kim, and Tian He. 2017. C-Morse: Cross-technology communication with transparent Morse coding. *INFOCOM* (2017), 1–9.
[28] Zhimeng Yin, Zhijun Li, Song Min Kim, and Tian He. 2018. Explicit Channel Coordination via Cross-Technology Communication. In *MobiSys* (Munich, Germany) *(MobiSys '18)*. Association for Computing Machinery, New York, NY, USA, 178–190. https://doi.org/10.1145/3210240.3210318