# EMU: Increasing the Performance and Applicability of LoRa through Chirp Emulation, Snipping, and Multiplexing

Fengxu Yang
yangfx@shanghaitech.edu.cn
SIST, ShanghaiTech Uni. & SARI, CAS
Shanghai, China

Pei Tian
tianpei2018@sari.ac.cn
SARI, CAS & UCAS
Shanghai, China

Xiaoyuan Ma
ma.xiaoyuan.mail@gmail.com
Shanghai Xuantu Intellig. Tech.
Shanghai, China

Carlo Alberto Boano
cboano@tugraz.at
Graz University of Technology
Graz, Austria

Ye Liu
liuye@must.edu.mo
Macau Uni. of Science & Technology
Macau, China

Jianming Wei
wjm@sari.ac.cn
SARI, CAS
Shanghai, China

## ABSTRACT

This paper presents EMU, a framework that enables the emulation, snipping, and multiplexing of LoRa chirps on commercial IoT devices equipped with low-power sub-GHz transceivers, including those supporting LoRa itself. Chirp snipping consists in artificially removing a sequence of chips and in putting the radio in low-power mode, which allows to reduce energy consumption while still communicating reliably. Chirp multiplexing exploits the gaps introduced by chirp snipping to transmit portions of another chirp on a separate channel, which allows to concurrently transmit two LoRa packets and to increase the throughput. We build EMU as a modular framework and implement support for off-the-shelf LoRa and non-LoRa transceivers. We then evaluate its performance by comparing the reliability, efficiency, and receiver sensitivity achieved by EMU with that of traditional LoRa for different physical layer settings. We finally showcase EMU's ability to send packets over two channels simultaneously, thereby improving the uplink throughput of LoRaWAN, and demonstrate that even non-LoRa transceivers employing EMU can communicate to a LoRaWAN gateway, enabling new use cases and expanding the applicability of LoRa technology.

## KEYWORDS

CC1125, Chirp, CSS modulation, Cross-technology communication, Emulation, Energy efficiency, IoT, LDRO, LoRa, LoRaWAN, LPWAN, Performance evaluation, Reliability, SX1276, Throughput.

## 1 INTRODUCTION

Low-power wide area networks (LPWANs) have recently become a key component of the Internet of Things (IoT), as they enable the interconnection of several devices over very large areas using energy-efficient and inexpensive radio transceivers [43, 46, 48]. In fact, thanks to current consumptions in the order of tens of mA and to communication ranges in the order of kilometers, LPWAN technologies such as LoRa, Sigfox, and Narrowband-IoT, drive the development of large-scale applications such as smart farming [62], industrial control [68], smart metering [16], air quality monitoring [6], smart transport [4], and water distribution management [13].

Thanks to its well-established ecosystem and low operating costs, LoRa is currently the most widespread and well-known LPWAN technology [22]. In contrast to solutions such as Sigfox and Narrowband-IoT, indeed, LoRa allows users to freely deploy their own network infrastructure and to maintain its full ownership and control without any subscription costs nor limitations on data traffic (besides national or regional duty-cycle constraints [52]).

Another key feature of LoRa contributing to its popularity is the high receiver sensitivity and resilience to interference thanks to the adoption of *chirp spread spectrum (CSS)* modulation. In fact, the carrier signal of LoRa consists of *chirps*, i.e., sweep signals whose frequency increases (up-chirps) or decreases (down-chirps) over time across a specific bandwidth (125, 250, or 500 kHz); data is then encoded by varying the starting frequency of these chirps [46]. The use of CSS modulation allows to detect and receive a signal even when its power is lower than the noise level, and to achieve data rates up to 11 kbps depending on the physical layer settings and local regulations [72]. Such data rates are much higher than the ones offered by competing technologies like Sigfox [44].

*The quest for improved performance and wider applicability.* Research on LoRa technology has surged in the last decade, fuelled by its growing popularity [63]. The community has relentlessly worked on increasing scalability [7, 36], energy efficiency [10, 11, 45, 76], and reliability [9, 14, 17, 42, 53], for example, by means of better coding schemes for data recovery [42, 53], enhanced link quality estimation techniques [17], as well as refined parametrization of physical layer settings [9]. Researchers have also derived mechanisms to increase data throughput (e.g., by leveraging concurrent transmissions [8, 39, 73]), and have investigated how to establish a communication with devices employing other technologies by means of cross-technology communication [38, 61], so to enable new use cases and push the applicability of LoRa even further.

**Contributions.** In this work[1], we aim to outstretch this body of literature with a solution that can increase the data throughput of LoRa-based systems, decrease their energy consumption, and enable new use cases that can expand the applicability of LoRa technology. We achieve all these goals by developing EMU, a framework that enables the emulation, snipping, and multiplexing of LoRa chirps on commercial IoT devices equipped with low-power sub-GHz transceivers (including those supporting LoRa itself). The operations of EMU stem from the observation that the CSS modulation adopted by LoRa results in a waveform consisting of short periods with increasing or reducing frequency, as shown in Fig. 3

[1]This work is partially funded by the National Science Foundation of China (No. 61902188).

and described in Sec. 2. One can hence emulate a LoRa signal by sending a carrier signal at a specific frequency for a short duration, so to generate an increasing or decreasing step function, where each step represents a LoRa chip.

*Saving energy via chirp snipping.* We find that LoRa nodes and LoRaWAN gateways can successfully decode a packet sent via chirp emulation even when transmitting incomplete chirps, i.e., when introducing artificial "gaps" in which the carrier signal is absent, so to artificially remove a sequence of chips. We show that this still yields a reliable communication, while introducing energy savings by up to 50% at the price of just a few dB lower receiver sensitivity.

*Throughput enhancement via chirp multiplexing.* Even more, we find that the gaps introduced in a chirp can be reused to transmit portions of another chirp on a separate channel, practically enabling the concurrent transmission of two LoRa packets that a LoRaWAN gateway can successfully decode at once. We implement this feature in EMU and show that it indeed allows to increase the data throughput by up to 105% when transmitting data to a gateway, while minimizing the energy expenditure even further.

*Automated open-source framework.* We build EMU as a modular framework that automatically calculates the characteristics of the emulated LoRa chirps for given payload lengths and physical layer settings (e.g., for a given spreading factor and coding rate configuration), and release its implementation to the public. Based on the computed frequencies and durations, each chip is generated in real time by activating the carrier transmission accordingly, and by snipping or multiplexing the chirps so to save energy and increase throughput. A hardware abstraction layer allows to keep the preparation and generation of LoRa waveforms independent from low-level platform details, which enhances portability.

*Support for off-the-shelf sub-GHz devices.* EMU runs on low-power sub-GHz radios supporting (G)FSK/OOK modulation and with a frequency resolution of at least 61 Hz (i.e., the minimum chip step in LoRa). Most LoRa transceivers fall in this category, as well as popular sub-GHz transceivers such as the Silicon Labs Si4463, or those supporting smart metering applications through the Wireless M-Bus protocol (e.g., the TI CC1125 and the Analog Devices ADF7030-1).

*Experimental evaluation.* After porting EMU on off-the-shelf LoRa (Semtech SX1276) and non-LoRa (TI CC1125) radios, we evaluate and showcase its benefits experimentally. We start by comparing the reliability, energy consumption, and sensitivity of LoRa and EMU on the SX1276 for different physical layer settings. We then showcase EMU's ability to send packets simultaneously over two channels, thereby improving the uplink throughput of LoRaWAN, and demonstrate that also non-LoRa transceivers employing EMU can successfully communicate to a LoRaWAN gateway.

**Paper outline.** After introducing background information about LoRa's channel coding and modulation in Sec. 2, we present EMU's main design principles (i.e., the chirp emulation, snipping, and multiplexing) in Sec. 3 and describe the framework's architecture and implementation in Sec. 4. We evaluate EMU's performance in Sec. 5 and showcase its benefits using a LoRaWAN gateway in Sec. 6. We finally summarize related work in Sec. 7, and conclude in Sec. 8.

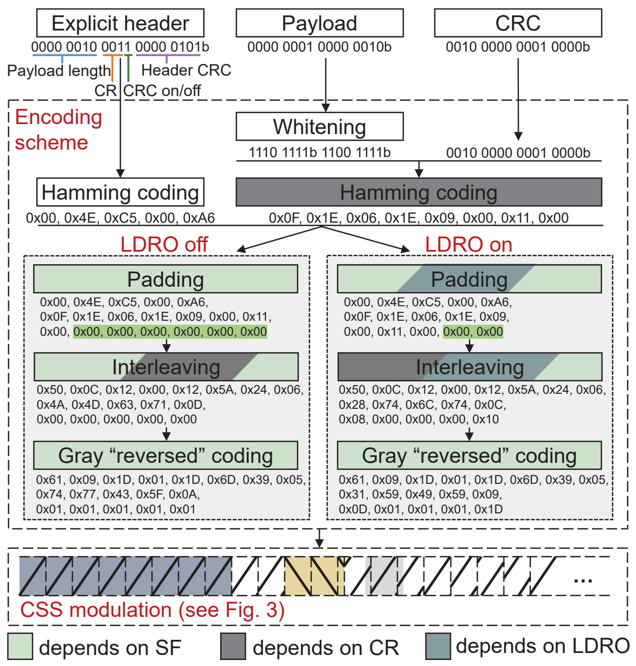

Explicit header
0000 0010 0011 0000 0101b
Payload length
CR
CRC on/off
Header CRC
Payload
0000 0001 0000 0010b
CRC
0010 0000 0001 0000b
Encoding scheme
Whitening
1110 1111b 1100 1111b
0010 0000 0001 0000b
Hamming coding
0x00, 0x4E, 0xC5, 0x00, 0xA6
Hamming coding
0x0F, 0x1E, 0x06, 0x1E, 0x09, 0x00, 0x11, 0x00
LDRO off
LDRO on
Padding
0x00, 0x4E, 0xC5, 0x00, 0xA6, 0x0F, 0x1E, 0x06, 0x1E, 0x09, 0x00, 0x11, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
Interleaving
0x50, 0x0C, 0x12, 0x00, 0x12, 0x5A, 0x24, 0x06, 0x4A, 0x4D, 0x63, 0x71, 0x0D, 0x00, 0x00, 0x00, 0x00, 0x00
Gray "reversed" coding
0x61, 0x09, 0x1D, 0x01, 0x1D, 0x6D, 0x39, 0x05, 0x74, 0x77, 0x43, 0x5F, 0x0A, 0x01, 0x01, 0x01, 0x01, 0x01
Padding
0x00, 0x4E, 0xC5, 0x00, 0xA6, 0x0F, 0x1E, 0x06, 0x1E, 0x09, 0x00, 0x11, 0x00, 0x00, 0x00
Interleaving
0x50, 0x0C, 0x12, 0x00, 0x12, 0x5A, 0x24, 0x06, 0x28, 0x74, 0x6C, 0x74, 0x0C, 0x08, 0x00, 0x00, 0x00, 0x10
Gray "reversed" coding
0x61, 0x09, 0x1D, 0x01, 0x1D, 0x6D, 0x39, 0x05, 0x31, 0x59, 0x49, 0x59, 0x09, 0x0D, 0x01, 0x01, 0x01, 0x1D
...
CSS modulation (see Fig. 3)
depends on SF
depends on CR
depends on LDRO

**Figure 1: Encoding scheme used by LoRa before transmission.** In this example, the payload consists of two bytes (`0x01` and `0x02`), and we make use of CR=1 and SF=7.

## 2 PRELIMINARIES

Before discussing how the emulation of LoRa chirps works and presenting the design of EMU, we introduce the reader to the encoding scheme used by LoRa (Sec. 2.1) and to CSS modulation (Sec. 2.2). We also shed light on LoRa's low data rate optimization (LDRO) used to increase the robustness of communications (Sec. 2.3). A thorough understanding of the LDRO is crucial in the design of EMU, as this feature is automatically enabled in LoRaWANs when using a high spreading factor. However, this feature is undocumented, so we had to first reverse-engineer it in order to generate emulated chirps that could be properly received by LoRaWAN gateways.

### 2.1 LoRa's Encoding Scheme

A LoRa transceiver builds a packet and encodes its data bits through a series of sequential steps, as shown in Fig. 1 and discussed next.

**Packet construction.** A LoRa packet starts with a preamble followed by an explicit header, a payload, and a 2-byte Cyclic Redundancy Check (CRC) computed based on the payload content. The length of the preamble sequence is set to 8 symbols in LoRaWAN (see Sec. 2.2 for more details). The explicit header has a length of 20 bits and contains information about the payload length, the employed coding rate, and whether the CRC is used.

**Whitening.** The first encoding step is the whitening of the payload, which consists in XORing each byte of data with a known pseudorandom sequence, and in swapping the two nibbles (groups of four bits). This allows to remove DC bias from the data and avoids the transmission of many consecutive 0s and 1s, which would cause the receiver to lose synchronization. In the example shown in Fig. 1, the payload bits `0x0102` are XORed with the sequence `0xFFFE`.

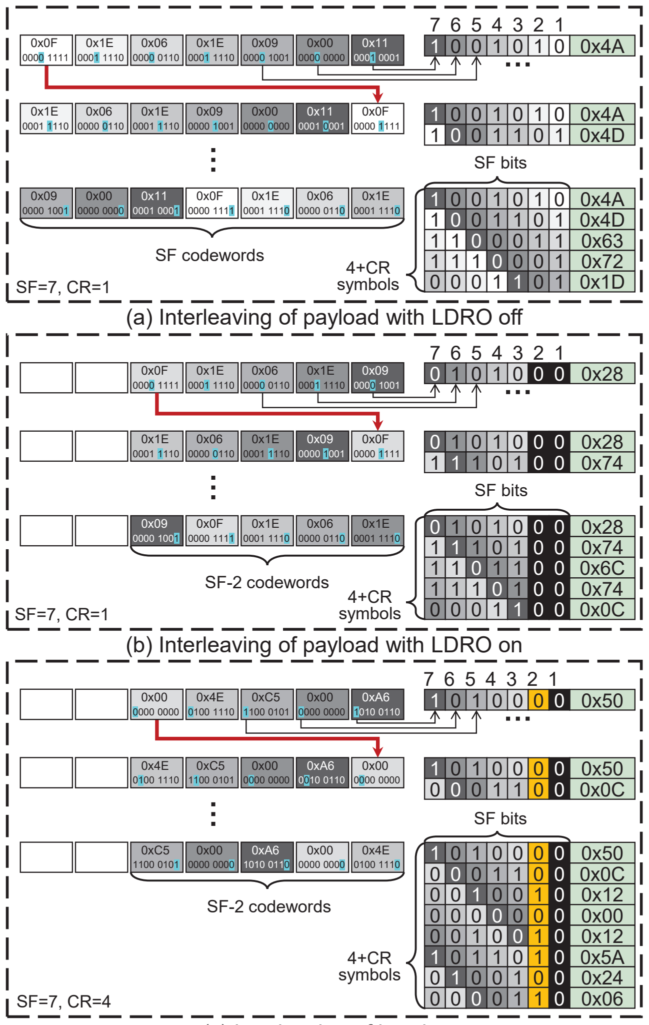


**Figure 2: Overview of the diagonal interleaving scheme used by LoRa when encoding data with and without the LDRO.**

**Hamming coding.** LoRa uses a variation of traditional Hamming codes to add redundancy in each codeword and enable error correction. In LoRa, the coding rate (CR) can range from 1 to 4, and a codeword consists of $4 + CR$ bits that are stored in one byte. The CR of both the payload and CRC is configurable and is set to 1 (i.e., a 1-bit code per nibble) in LoRaWAN. The 5 bits originating from each nibble are preceded by three zeros to form a byte. The CR of the explicit header is fixed to 4 (i.e., a 4-bit code per nibble). In the example shown in Fig. 1, the 13-nibble packet is transformed into 13 codewords (each 1-byte long) after applying Hamming coding.

**Interleaving.** To limit the impact of bursty noise and fading to a single bit error per symbol, LoRa scrambles data bits throughout a packet (i.e., it spreads the bits constituting a codeword between multiple symbols) by means of a process called diagonal interleaving [51, 66], which is illustrated in Fig. 2. Interleaving is performed on a number of codewords that is proportional to the spreading factor (SF) used when modulating data, where $SF \in \{7...12\}$ and $N = 2^{SF}$ chips per symbol are used by LoRa's CSS modulation. In practice, a bit matrix with SF columns and $(4 + CR)$ rows is derived, where the first row is obtained by picking the $(4 + CR)^{th}$ bit in each codeword, and every following row, the $i^{th}$ row, picks the $(4 + CR - i)^{th}$ bit in each codeword, and performs a 1-codeword cyclic left shifting, as shown by the red arrows in Fig 2. Each row in the matrix corresponds to a symbol sent using CSS modulation after applying Gray "reversed" coding. In the example shown in Fig. 2 (a), SF=7 and CR=1, meaning that the bit matrix obtained when interleaving the payload is composed of 5 rows and derived from 7 codewords. When interleaving the header (Fig. 2 (c)), however, one always employs CR=4 and uses SF-2 codewords (the lowest 2 bits of each row are fixed to zeros), which results in a matrix with 8 rows. The column marked in yellow is computed as the parity of all 7 bits.

**Padding.** Since interleaving is performed on a given number of codewords (SF or $SF - 2$), one may need to perform bit padding by adding extra bits (`0x00`) at the end of data portion. In the example shown in Fig. 1, the header has a number of codewords that is a multiple of 5 (i.e., SF-2). The remaining part should have a number of codewords that is a multiple of 7 (i.e., SF used for transmission): thus, six codewords (highlighted in green) were added to the payload.

**Gray "reversed" coding.** The output of the interleaving stage is grouped by SF bits and mapped to LoRa symbols (which can be seen as integers between 0 and $2^{SF} - 1$) by using Gray "reversed" coding. In the Gray representation, adjacent symbols only differ by one bit. This property increases the robustness of the decoding process when a symbol is more likely to be misinterpreted as an adjacent one (rather than misinterpreted as a random one) due to carrier and sampling frequency offsets [27]. In fact, a symbol being mistaken for an adjacent one only causes a single bit error, which can be corrected by Hamming codes [66, 67].

## 2.2 LoRa's CSS Modulation

LoRa is a spread-spectrum frequency modulation using a bandwidth $BW \in \{125, 250, 500\}$ kHz and $N = 2^{SF}$ chips per baseband symbol. A symbol, which can be seen as an integer number $s \in \{0, ..., N-1\}$, spans throughout $BW$ and resembles a step function, as shown in Fig. 3. Specifically, a symbol starts at $\frac{s \cdot BW}{N} - \frac{BW}{2}$ and its frequency increases by $BW/N$ in every chip, whose duration is $1/BW$. A frequency folding to $-BW/2$ occurs when reaching $BW/2$, and the increase in frequency by $BW/N$ continues in every following chip until the initial frequency is reached again [67]. A symbol corresponds to a chirp: chirps with increased and decreased frequency are defined as up-chirps and down-chirps, respectively.

In the example shown in Fig. 3, which shows the chirps generated by LoRa when transmitting the packet with payload `0x0102` presented in Fig. 1 with SF=7 and BW=125 kHz, the frequency step size is $BW/N = 976.5$ Hz. The initial frequency of symbol $s =$ `0x09` is $\frac{s \cdot BW}{N} - \frac{BW}{2} = -53710.9$ Hz. In addition, the length of the symbol can be represented as $2^{SF}/BW$, meaning that decreasing the bandwidth or increasing the spreading factor will both increase the length of the symbol, which brings similar effects on the sensitivity [7].

**Waveform generation.** In addition to the encoded symbols derived from the explicit header, payload, and CRC, an 8-symbol *preamble*, a 2-symbol *sync word*, and a 2.25-symbol *down-chirp* are transmitted. The preamble, composed of eight up-chirps starting at $-BW/2$, is

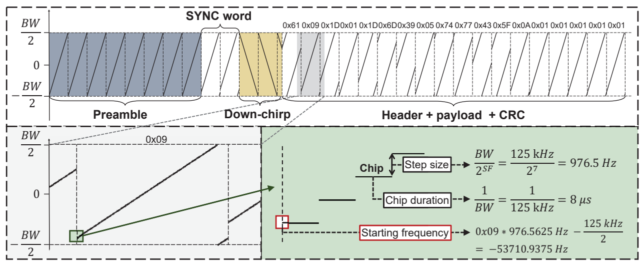

SYNC word
0x61 0x09 0x1D 0x01 0x1D 0x6D 0x39 0x05 0x74 0x77 0x43 0x5F 0x0A 0x01 0x01 0x01 0x01 0x01
$\frac{BW}{2}$
0
$-\frac{BW}{2}$
Preamble
Down-chirp
Header + payload + CRC
0x09
Chip
Step size
$\frac{BW}{2^{SF}} = \frac{125\ kHz}{2^7} = 976.5\ Hz$
Chip duration
$\frac{1}{BW} = \frac{1}{125\ kHz} = 8\ \mu s$
Starting frequency
$0x09 * 976.5625\ Hz - \frac{125\ kHz}{2} = -53710.9375\ Hz$

**Figure 3: Waveform transmitted by LoRa to send the encoded packet from Fig. 1.** SF=7, BW=125 kHz, and the LDRO is disabled.

used to wake up receivers. The sync word, also composed of up-chirps, indicates whether public LoRaWANs or private networks are used. The 2.25-symbol down-chirp starts at $BW/2$ and is used to calibrate the carrier frequency and symbol timing at the receiver. For low data rates, the LDRO is recommended [57, 59], and it is automatically enabled by LoRaWAN devices when using a high SF. We provide additional details about this option next.

## 2.3 LoRa's Low Data Rate Optimization (LDRO)

The LDRO allows to relax the oscillator requirements and to improve the packet reception ratio (PRR) when a symbol lasts for more than 16 ms, e.g., when using a BW of 125 kHz and a SF of 11 or 12 [59]. This optimization can be enabled manually by simply configuring a bit in the corresponding register (e.g., the `RegModemConfig3` in the `SX1276` radio [59]) before transmission. In our experiments, enabling the LDRO when sending a 51-byte LoRa packet with SF=12 helps increasing the PRR from 65.4% to 99.9%: this is why LoRaWAN gateways enable this option by default when using SF 11 or 12.

This feature, however, is only mentioned in a few documents [3, 33, 55], and no *complete* description is available. As we aim to support CSS emulation over *all* possible SFs and allow communication with *off-the-shelf* devices, the lack of documentation poses a great challenge. In fact, both transmitter and receiver need to have an identical LDRO configuration: a receiver with LDRO disabled cannot decode a waveform sent with the LDRO enabled, and vice-versa. Hence, if we just disable the LDRO, we would be unable to communicate with LoRaWAN gateways using SF 11 or 12.

For this reason, we performed some reverse engineering to better understand how this feature works. We find that the padding and interleaving stages in the encoding scheme presented in Sec. 2.1, as well as the computation of the starting frequency of a chirp presented in Sec. 2.2 are slightly different when enabling the LDRO.

**Changes in interleaving.** When the LDRO is enabled, each symbol still consists of SF bits, but the lowest two bits are fixed to zeros. Therefore, similar to the interleaving of the header, only SF-2 codewords are used when performing diagonal interleaving. The difference is that only zeros are set (the lowest two columns in Fig. 2 (b)) and no parity is computed in the second lowest bit (as was instead done for the header, see the yellow column in Fig. 2 (c)). When using only the highest 5 bits for interleaving, the minimum difference between symbols increases from 1 to 4, which leads to a higher reliability. Fig. 2 (b) shows an example of the interleaving stage when enabling the LDRO while adopting SF=7 and CR=1.

**Changes in padding.** Because of the aforementioned changes in the interleaving, the padding stage needs to generate a number of codewords that is a multiple of $SF - 2$ when the LDRO is enabled. For instance, when using SF=7 with the exemplary 2-byte payload shown in Fig. 1, only two bytes need to be padded instead of six.

**Changes in frequency calculation.** Even though the minimum difference between symbols increases to 4 when using only the highest 5 bits for interleaving, after applying Gray "reversed" coding, such difference may still be only 1. Consider, for instance, having symbols `0x1c` and `0x14` as output of the interleaving stage: their relative difference is 8. However, after applying Gray "reversed" coding, we obtain symbols `0x18` and `0x19`, which only differ by 1. For this reason, when the LDRO is enabled, even symbols are pre-processed and "3" is subtracted from their value: this way, `0x18` minus three becomes `0x15`, which differs from `0x19` by 4 units. During CSS modulation, the start frequency of each chirp is calculated based on the value of the pre-processed symbols. This results in a coarser starting frequency which can be discriminated more easily at the receiver, thereby relaxing the clock drift requirements.

**Overhead.** Given the same physical layer settings, LDRO-enabled packets embed more symbols. According to [59], for a given preamble length, the number of symbols $n$ can be calculated as:

$$n_{packet} = n_{preamble} + 4.25 + n_{payload} \tag{1}$$

$$n_{payload} = 8 + max\left(\left\lceil \frac{8PL - 4SF + 28 + 16CRC - 20IH}{4(SF - 2DE)} \right\rceil (CR + 4), 0\right) \tag{2}$$

where $PL$ represent the number of payload bytes; $CR$ is coding rate; $IH$ indicates the use of an explicit (0) or implicit (1) header; $DE$ indicates whether the LDRO is enabled (1) or disabled (0). This shows how $n_{payload}$ increases when LDRO is enabled, resulting in less effective bits carried by each symbol, as well as longer transmissions.

## 3 EMU: DESIGN PRINCIPLES

In this section, we describe the three key functions underpinning the design of EMU[2]: chirp emulation, snipping, and multiplexing. With EMU, one can *emulate* chirps, i.e., LoRa's CSS waveforms, by generating a frequency-varied carrier using an off-the-shelf (G)FSK/OOK modulator (Sec. 3.1). Furthermore, the emulated chirps can be *snipped* to decrease the radio-on time and save energy (Sec. 3.2), or even *multiplexed* in time to enable the parallel transmission of two messages and increase data throughput (Sec. 3.3).

## 3.1 Chirp Emulation

Practically, a chirp can be regarded as a sequence of chips at increasing or decreasing frequencies, where a chip is a short sine waveform sent at a specific frequency, as illustrated in Fig. 4 (top-left). Because of this, the various chips can be generated by transmitting a carrier signal, and one can emulate chirps if the carrier frequency can be updated as soon as the next chip arrives (i.e., at the time indicated by the vertical dashed lines shown in Fig. 4 top-left).

**Pre-requisites.** Regardless of the SF, in LoRaWANs [41], the duration of a chip is $1/BW = 8\ \mu s$ when having $BW = 125$ kHz. The

[2]Our **emu**lation framework shares its name with the second-largest living bird on Earth by height, which – as LoRa signals – can travel very long distances. Distinctive feature of emus are their feathers, as they have a **double** plume (i.e., two feathers come out of one shaft) with sharply outlined barbs, which recalls the ability of our framework to transmit concurrently two packets by means of chirp multiplexing [49].

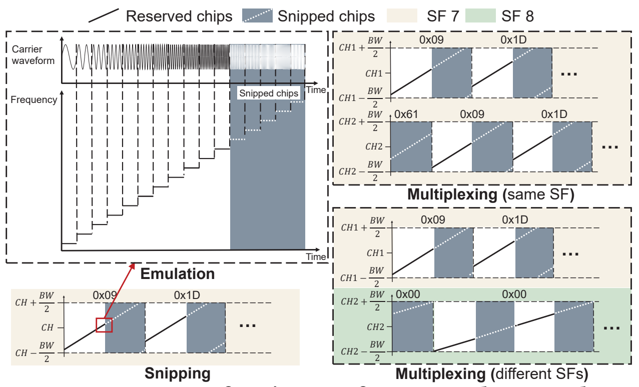


**Figure 4: Overview of EMU's main functions: chirp emulation (top-left), snipping (bottom-left), and multiplexing (right).** Chirp multiplexing supports also two waveforms with different SF.

chip step is $BW/N = 976.5$ Hz when using SF=7, and 122.1 Hz when using SF=12 with the LDRO enabled, as illustrated in Fig. 3. A CSS waveform can hence be emulated by generating a carrier signal for short periods under two key pre-requisites:

(i) One can configure the carrier frequency at least every 8 $\mu$s;
(ii) One can configure the carrier frequency with a resolution that is sufficient to discriminate adjacent chips (the worst case is represented by SF=12, with a chip step of 122.1 Hz).

In most off-the-shelf sub-GHz radio transceivers, one can transmit a carrier with a configurable frequency using a (Gaussian) Frequency Shift Keying ((G)FSK) or On-Off Keying (OOK) modulator. In an OOK modulator, the absence or presence of a carrier are represented by bits 0 and 1, respectively. In a (G)FSK modulator, a carrier always exists, and digital information is transmitted through discrete frequency changes of the carrier signal. Hence, both (G)FSK and OOK modulators can generate a carrier, and typically have two operation modes: *packet-oriented* and *bit-oriented*. In packet-oriented mode, it is difficult to achieve a fine-grained chirp emulation, since transmitting the shortest packet (i.e., with a payload length of zero) still leads to an on-air time of tens of microseconds, which is above our 8 $\mu$s pre-requisite. Conversely, the use of the bit-oriented mode allows more flexibility, as one can start/stop the transmission of the carrier by setting/clearing a dedicated pin of the transceiver. One can also switch frequency at a relatively arbitrary point in time, which allows to meet the first pre-requisite. We provide details on how we make use of a transceiver's bit-oriented mode in Sec. 4.3.

The data-sheet of most off-the-shelf sub-GHz radios specifies the *frequency resolution*, i.e., how finely one can set the carrier frequency. This info can be used to determine whether the (G)FSK/OOK modulator can fulfill the second pre-requisite, and generate adjacent chips that can be discriminated. Specifically, a chip step can be discriminated as long as the frequency resolution is *less than half of the chip step*. This corresponds to 488.28 Hz for SF=7, and 61.05 Hz for SF=12, with the latter being the worst-case: the radio hence needs to have a frequency resolution *below* 61.05 Hz. Fortunately, most off-the-shelf sub-GHz transceivers meet this requirement. Examples are Semtech's SX1276 and SX1208, Texas Instruments' CC1125 and CC1200, as well as Silicon Labs' Si4463 and Si4464.

**Principle.** If the aforementioned pre-requisites are met, the actual chirp emulation simply consists in configuring the OOK/(G)FSK modulator to transmit a '1' bit, so to activate the generation of a carrier, and in computing and adjusting the frequency over time. This step can be accomplished at the end of each chip: we will explain this in more detail in Sec. 4.1. As discussed in Sec. 2.2, when emulating up-chirps, we perform a frequency folding to $-BW/2$ when reaching $BW/2$, and a frequency folding to $+BW/2$ when reaching $-BW/2$ during the emulation of down-chirps.

## 3.2 Chirp Snipping

One of the key features of EMU is the ability to reduce energy consumption by means of chirp snipping.

**Principle and feasibility.** The principle behind chirp snipping is illustrated in Fig. 4 (bottom-left): a chirp can still be decoded successfully even when a portion of its chips are not transmitted (e.g., when the portion of chips with gray background are omitted). Next, we prove the feasibility of snipping chirp through theoretical analysis. An encoded up-chirp and down-chirp can be described as:

$$C_{sym}(t, f_s) = e^{j2\pi(-\frac{BW}{2}+\frac{1}{2}kt)t} \cdot e^{j2\pi f_s t} = C(t) \cdot e^{j2\pi f_s t} \tag{3}$$

$$C^{\S}(t) = C_{downchirp}(t) = e^{j2\pi(\frac{BW}{2}-\frac{1}{2}kt)t} \tag{4}$$

where $f_s$ and $BW$ identify the start frequency of a symbol and the bandwidth, respectively. $C(t)$ and $C^{\S}(t)$ represent the raw up-chirp and down-chirp. A snipped symbol can be represented as follows:

$$C_{snipped}(t, f_s) = \begin{cases} 0 & t_{\text{snip\_start}} < t \le t_{\text{snip\_end}}, \\ C_{sym(t,f_s)} & \text{otherwise.} \end{cases} \tag{5}$$

A chirp is snipped between $t_{\text{snip\_start}}$ and $t_{\text{snip\_end}}$. A receiver would de-chirp first and then extract the dominant frequency component based on the de-chirped signal. Assuming the symbol is aligned, de-chirping can be regarded as a product of the received snipped up-chirp and a raw down-chirp [79]:

$$\begin{aligned} C^{\S}(t) * C_{snipped}(t, f_s) &= \begin{cases} 0 \\ C^{\S}(t) * C(t) \cdot e^{j2\pi f_s t} \end{cases} \\ &= \begin{cases} 0 & t_{\text{snip\_start}} < t \le t_{\text{snip\_end}}, \\ e^{j2\pi f_s t} & \text{otherwise.} \end{cases} \end{aligned} \tag{6}$$

We can notice that the snipped signal after de-chirping still contains $f_s$ for demodulation, which represents the value of a symbol and can be extracted after a fast Fourier transform. Moreover, the snipping position remains arbitrary, indicating that the receiver can still extract the value of a symbol regardless of how the chirp was snipped. It should also be noted that the peak of the frequency is lower than the standard CSS signal, since a symbol has lost part of its energy. Hence, the SNR is lower than that of the default CSS signal, which may result in a sensitivity loss (as shown in Sec. 5.3).

**Creating a gap.** Now that we have shown that LoRa snipped chips can still be demodulated by the receiver, we discuss how one can implement such functionality on real hardware. We call the time in which chirps are not transmitted a *gap*: the introduction of the latter allows to put the radio transceiver in low-power mode and save energy. Chirp snipping can be implemented by transmitting a '0' bit when making use of an OOK modulator: this turns off the radio's power amplifier (PA), resulting in a lack of carrier. One can further reduce the current draw of the transceiver during gaps by

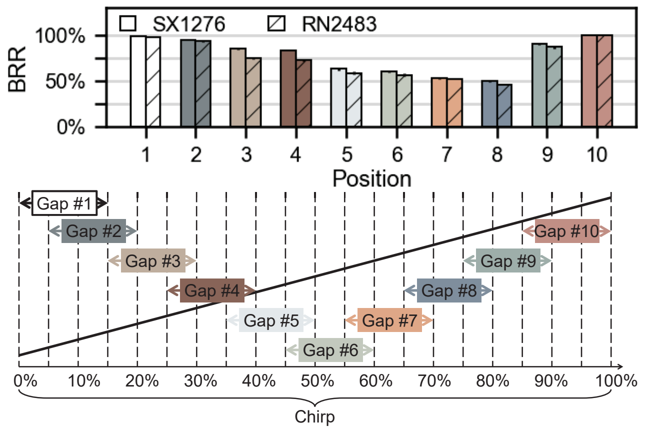


**Figure 5: BRR as a function of the gap position when snipping the chirps with different LoRa devices.** The best performance can be obtained when snipping the chips towards the end.

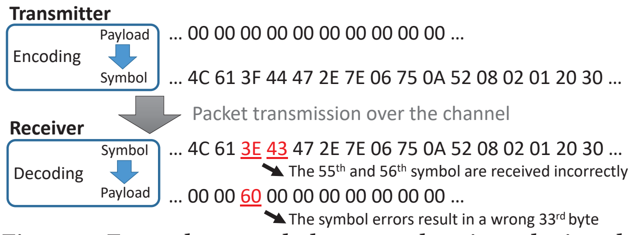

Transmitter
Encoding
Payload
Symbol
... 00 00 00 00 00 00 00 00 00 00 ...
... 4C 61 3F 44 47 2E 7E 06 75 0A 52 08 02 01 20 30 ...
Packet transmission over the channel
Receiver
Decoding
Symbol
Payload
... 4C 61 3E 43 47 2E 7E 06 75 0A 52 08 02 01 20 30 ...
The 55th and 56th symbol are received incorrectly
... 00 00 60 00 00 00 00 00 00 00 ...
The symbol errors result in a wrong 33rd byte

**Figure 6: Exemplary symbol errors when introducing the gap in the middle of the chirps.** We observe mostly a few non-consecutive errors caused by symbol misalignment. This hints that a reduced performance of the receiver's adaptive symbol alignment mechanism may be causing the low BRR observed in Fig. 5.

switching its status from *transmit mode* to *standby mode*, which is a common feature of off-the-shelf sub-GHz radios [59] (also known as *idle mode* in Texas Instruments' CCxx series [69, 70]). In standby mode, typically only the crystal oscillator remains active, which gives the possibility to retrieve some basic timing information. At the end of a gap, the carrier transmission can be reactivated by re-entering *transmit mode* and by transmitting a '1' bit. When employing a (G)FSK modulator, instead, one can perform chirp snipping by directly entering *standby mode*. We describe in detail how to carefully control the timing of the snipping in Sec. 4.3.

**Where to create a gap?** Fig. 4 (bottom-left) shows that EMU snips the chips towards the end of a chirp. This is because our experiments have shown that this is the most reliable configuration, allowing a LoRa packet to still be correctly demodulated. In fact, if the gap is introduced elsewhere in the chirp (e.g., in the middle of the chirp), the performance would degrade significantly. Fig. 5 shows the byte reception rate (BRR) between a pair of nodes measured as a function of the gap position. We obtain these results by transmitting 60000 LoRa packets using chirp emulation, while snipping 15% of the chirps using different LoRa platforms, namely the Semtech SX1276 and the Microchip RN2483. Each packet has a length of 255 bytes, and the content of its payload is randomized; the SF and CR are set to 7 and 1, respectively, and the LDRO is disabled. Despite the nodes being in close proximity (10 cm), the introduction of a gap in the middle of the chirp causes the BRR to drop to 50% with both platforms. The best performance is achieved when the gap is introduced at the beginning and at the end of the chirp, with a slightly higher BRR in the latter case: for this reason, we design EMU to snip the end of a chirp. We believe that the reason for this is hardware-dependent and related to the adaptive symbol alignment mechanism of the receiver. As the implementation of the transceivers is closed-source, we substantiate our hypothesis with empirical observations. Fig. 6 shows a packet filled with zeros transmitted while introducing a gap in the middle of the chirps. We observe the received symbol errors by exploiting the encoding scheme introduced in Sec. 2.1 and find that: (i) the values of the erroneous received symbols are very close to the actually-transmitted values, which hints that the errors are due to symbol misalignment; (ii) the symbol errors are not consecutive, which hints the existence of an adaptive mechanism to fine-tune the alignment for each symbol at the receiver (i.e., symbols can be received correctly again soon afterwards, once the symbol alignment is recovered by the receiver). We will investigate in future work whether our hypothesis is correct and why the adaptive symbol alignment performs poorly when the gap is introduced in the middle of the chirps.

**Trade-off energy vs. robustness.** Chirp snipping is feasible in practice because a LoRa waveform is meant to resist bursty interference lasting for up to 30% of a symbol duration [54], i.e., a symbol can still be decoded successfully even when receiving only a part of it. We find indeed that one can snip more than 30% of a chirp while still sustaining a PRR of 100%, which allows to strongly reduce the energy consumption when transmitting data (see Sec. 5 and 6). EMU performs chirp snipping on the entire LoRa packet, except for the preamble, the sync word, and the down-chirps. This way, the snipping does not affect the packet detection and the clock calibration at the receiver. Doing so also provides a higher robustness to interference: Haxhibeqiri et al. [28] have shown that packet loss in LoRa mostly occurs when interference hits the preamble and is almost zero when the payload is hit. Still, introducing gaps in the chirps causes a reduction in receiver sensitivity. Our experiments show, for example, that the sensitivity decreases by up to 9 dBm, i.e., from -128 dBm to -119 dBm, when using SF=12 and snipping 65% of a chirp (see Sec. 5.3). While this may look undesirable, a sensitivity of -119 dBm still suffices to transmit data over large distances (up to 2700 m), which is sufficient for most LoRa-based applications. The introduction of gaps in the chirps also causes a higher vulnerability to noise. If an interference source $N(t)$ is present during transmission, the de-chirped waveform would be $e^{j2\pi f_s t} + C(t) * N(t)$. Therefore, following Eq. 6, if interference occurs while sending a snipped chirp, $C^{\S}(t) * C_{snipped}(t, f_s)$ would no longer be zero when $t_{\text{snip\_start}} < t \leq t_{\text{snip\_end}}$. This increases the likelihood that the dominant frequency component after demodulation is affected, possibly leading to an incorrect reception. An evaluation of the system performance in the presence of symbol collisions goes beyond the scope of this paper and will be addressed in future work.

## 3.3 Chirp Multiplexing

Chirp snipping allows to save radio-on time by generating a gap within chirps, i.e., by keeping the (G)FSK/OOK modulator silent instead of transmitting a sequence of chips. Chirp multiplexing exploits the gap created within a chirp to send another (different) chirp on another RF channel (with the same or different SF), practically enabling the concurrent transmission of two LoRa packets sharing the same preamble at once. Therefore, chirp multiplexing allows to increase throughput: this is especially important given the capability of LoRaWAN gateways of receiving multiple packets from different channels simultaneously. EMU allows to multiplex also identical chirps, which can be a useful feature to send high-priority packets on two channels in parallel to increase the probability of successful reception by the gateway.

**Principle.** Also chirp multiplexing is inspired by the observation that the reception of a partial chirp is sufficient for correct decoding. Fig. 4 (top-right) illustrates the basic idea behind chirp multiplexing when sending two identical chirps with the same SF. The transmission of the first half of a chirp in a first RF channel (CH1) is replicated on a second channel (CH2) during the gap created in the second half of the first chirp. The two chirps need to be sent on different RF channels in order to avoid interference at the receiver: our experiments show that a difference of at least 400 kHz is sufficient for this. When multiplexing two chirps with different SFs, one needs to select adjacent SFs (e.g., SF 7 and 8): if one would pick very different configurations (e.g., SF 7 and 12), the chirp with a greater SF is interrupted too often, which would compromise its reception. Fig. 4 (bottom-right) shows an example of chirp multiplexing when sending different chirps with different SF (7 and 8, respectively). EMU always makes sure to transmit the first portion of a chirp: only then it introduces a gap (i.e., it transmits the first half of the chirp with SF=7, and the first quarter of the chirp with SF=8): this ensures a reliable detection at the receiver, as discussed in Sec. 3.2.

## 4 EMU: ARCHITECTURE & IMPLEMENTATION

As the three key functions of EMU are now clarified, we discuss next its architecture and implementation in detail. EMU is conceived as a modular platform that abstracts low-level details and is independent from hardware-specific features. We implement support for a LoRa transceiver (the Semtech SX1276), as well as for a non-LoRa chip (the TI CC1125), and release our code to the public[3], so to contribute to further developments in the ever-growing LoRa community. After giving an overview of EMU's architecture (Sec. 4.1), we provide a detailed description of the module responsible for the waveform preparation (Sec. 4.2) and generation in real-time (Sec. 4.3).

### 4.1 Overview of EMU's Modular Architecture

Fig. 7 shows EMU's architecture, which is composed of 3 main layers. The *abstract layer* acts as an interface to the application in order to exchange information about the content (i.e., explicit header, payload, CRC) and configuration (i.e., SF, CR, LDRO, TX power, etc.) of the LoRa packets to be sent. It also selects the mode with which the waveform should be generated (i.e., simple chirp emulation, chirp snipping, or chirp multiplexing). In case chirp multiplexing is adopted, information about a second LoRa packet and its related parameters (marked as * in Fig. 7) also needs to be provided.

The *waveform preparation layer* receives the information from the *abstract layer* and encodes the packet into a sequence of symbols as

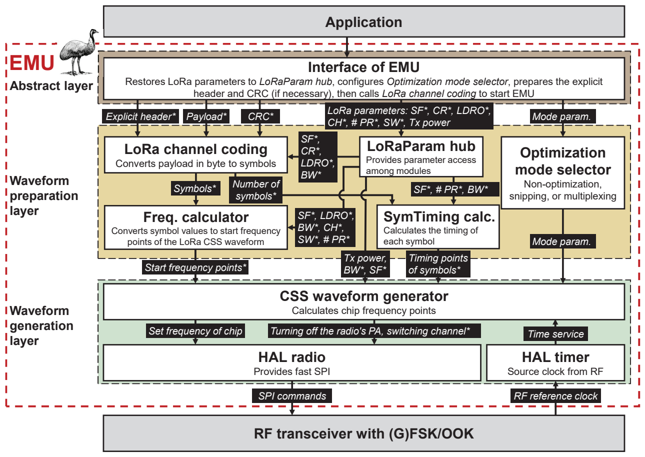

Application
EMU
Abstract layer
Interface of EMU
Restores LoRa parameters to LoRaParam hub, configures Optimization mode selector, prepares the explicit header and CRC (if necessary), then calls LoRa channel coding to start EMU
Explicit header*
Payload*
CRC*
LoRa parameters: SF*, CR*, LDRO*, CH*, # PR*, SW*, Tx power
Mode param.
Waveform preparation layer
LoRa channel coding
Converts payload in byte to symbols
SF*, CR*, LDRO*, BW*
LoRaParam hub
Provides parameter access among modules
Optimization mode selector
Non-optimization, snipping, or multiplexing
Symbols*
Number of symbols*
SF*, # PR*, BW*
Freq. calculator
Converts symbol values to start frequency points of the LoRa CSS waveform
SF*, LDRO*, BW*, CH*, SW*, # PR*
SymTiming calc.
Calculates the timing of each symbol
Mode param.
Start frequency points*
Tx power, BW*, SF*
Timing points of symbols*
Waveform generation layer
CSS waveform generator
Calculates chip frequency points
Set frequency of chip
Turning off the radio's PA, switching channel*
Time service
HAL radio
Provides fast SPI
HAL timer
Source clock from RF
SPI commands
RF reference clock
RF transceiver with (G)FSK/OOK

**Figure 7: Overview of EMU's architecture.** CH indicates the channel frequency, SW the sync word, and # PR the number of preamble chirps. One more payload and its parameters (with an asterisk) are passed when performing chirp multiplexing.

described in Sec. 2.1 (LoRa channel coding). It then computes the starting frequency (Freq. calculator) and the timing (SymTiming calc.) of each symbol based on the chosen SF and BW.

The *waveform generation layer* converts the symbols provided by the upper layer into a sequence of chips (CSS waveform generator), and transmits them in real-time by controlling the radio through a hardware abstraction layer (HAL radio) module that allows to configure the frequency of the carrier. When chirp snipping or multiplexing is adopted, the CSS waveform generator module would also instruct the HAL radio to switch channel (for multiplexing) or to turn off the radio's PA and switch to standby mode (for snipping).

**Memory footprint.** The binary size of EMU on the SX1276 is 36.8 kB, and the RAM usage is 18.9 kB. The *waveform preparation layer* is the module responsible for the highest RAM consumption (i.e., 87% out of 18.9 kB), the reason being that the computation of the starting frequency and timing of symbols needs to be performed in advance to ensure a real-time generation of the waveform.

### 4.2 Waveform Preparation Layer

The implementation of the encoding scheme used by LoRa in EMU's *waveform preparation layer* (LoRa channel coding module) is inspired by the open-source LoRa PHY prototype for GNU radios[4] developed by Tapparel et al. [67]. This implementation, however, lacks LDRO support, which is a critical function when employing a SF of 11 or 12, since LoRaWAN gateways enable this feature by default. Because the exact operating principle of the LDRO is undocumented, we conducted extensive reverse-engineering to shed light on how to correctly encode data when this option is enabled. Our findings have already been described in Sec. 2.3 and have been integrated into the LoRa channel coding module) accordingly. The obtained sequence of symbols are then fed to the Freq. calculator and SymTiming calculator modules, which compute the starting frequency and timing of each symbol (as well as for the preamble, sync word, and for the 2.25-symbol down-chirp). Note that the

[3]https://github.com/sari-wesg/emu

[4]https://github.com/tapparelj/gr-lora_sdr, commit: 1354b51.

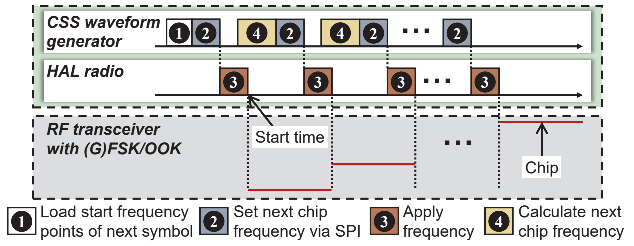

CSS waveform generator
HAL radio
RF transceiver with (G)FSK/OOK
Start time
Chip
1 Load start frequency points of next symbol
2 Set next chip frequency via SPI
3 Apply frequency
4 Calculate next chip frequency

**Figure 8: Timing diagram of chirp waveform generation.**

LoRaParam hub module is designed for storing the configured LoRa parameters (e.g., SF, BW, CR, and LDRO) and for providing access to them from other modules, thereby decoupling the operations of individual modules from application-specific parameters.

### 4.3 Waveform Generation Layer

The *waveform generation layer* is in charge of activating/deactivating the carrier and of configuring its frequency over time, so to emulate a chirp chip-by-chip. While a chip is being generated, this layer already computes the frequency of the next chip, and starts instructing the radio to update the carrier frequency by writing the corresponding registers, so to ensure a timely generation.

Fig. 8 illustrates the various interactions across the CSS waveform generator and HAL radio modules. First, the micro-controller loads the starting frequency of the next symbol provided by the upper layer ❶. This starting frequency is communicated to the HAL radio via SPI ❷. At this instant, the CSS waveform generator retrieves the exact radio timing from the HAL timer and regards this timestamp as zero[5]. While the HAL radio applies the change in frequency and the chip starts being generated ❸, the next chip frequency is calculated by the CSS waveform generator ❹; the latter will then inform the HAL radio about the change in frequency via SPI. This sequence of operations iterates until all chips of the chirp have been generated. To minimize delays, all float multiplications are replaced with fixed point and shift operations.

The timing information provided by the HAL timer is used to compute the chip duration, and is hence critical for the success of chirp emulation. Any drift of the timer would in fact lead to inaccuracies in the symbol duration and trigger an error on the receiver. Since the chip duration is 8 $\mu$s for $BW = 125$ kHz, a $\mu$s-level time granularity is required, meaning that the source clock of the timer should not be lower than 1 MHz. In off-the-shelf platforms like ChirpBox [73], which embed an STM32L476RG micro-controller and a Semtech SX1276 LoRa transceiver, the micro-controller only connects to an external crystal of 32 kHz. Luckily, the SX1276 transceiver can output a 1 MHz clock signal (CLKOUT): one can hence select this output, which has an accuracy of 20 ppm [59], as the timer source clock[6]. Note that in the STM32L476RG micro-controller used by ChirpBox, only 16-bit timers are available, which means that a timer overflows every 65.5 ms. As this is insufficient to keep track of the time needed to send an entire LoRa waveform (which may last for several seconds when using high spreading factors), we chain two timers internally, practically obtaining a 32-bit timer.

[5]The actual instant in which the waveform starts being generated occurs slightly later, due to SPI delays and the lag caused by the radio when applying the frequency change.
[6]A similar pin provided by radio transceivers like the TI CC1125 is called SERIAL_CLK. In case no such pin is available, an internal clock has to be activated as the timer source clock and calibrated periodically with the external 32 kHz crystal.

**Architectural support for EMU.** With the ever-decreasing cost of micro-controllers, wireless embedded systems may soon be equipped with a co-processor handling timing-sensitive radio tasks in a dedicated fashion [34, 64, 65]. Although EMU is currently implemented as an embedded software framework, it can evolve as a co-processor subsystem: in such an architecture, the main processor can call all EMU's functions to run in the co-processor via a general hardware interface, e.g., an SPI channel with a clock $\geq$ 10 MHz and a few I/O pins. In order to timely adjust the frequency of the carrier and to accelerate the computation of the frequency points in the *waveform generation layer*, the co-processor should ideally have an 80 MHz clock and embed a hardware floating-point unit.

## 5 EVALUATION

We evaluate the performance of EMU experimentally, and investigate its reliability, energy efficiency, and receiver sensitivity. Specifically, our evaluation answers the following questions:

- How reliable are EMU's main features, i.e., simple chirp emulation, chirp snipping, and chirp multiplexing? (Sec. 5.1)
- For how long can a chirp be snipped? What is the relationship between the amount of chips snipped and the ability to successfully decode a message? (Sec. 5.1)
- How energy efficient is EMU? How much energy can be saved thanks to chirp snipping and multiplexing? (Sec. 5.2)
- How does EMU's receiver sensitivity compare to that of traditional LoRa? Do chirp snipping and multiplexing affect the receiver sensitivity significantly? (Sec. 5.3)

The results shown in this section are based on the SX1276 platform: EMU is used when transmitting data, whereas a classic LoRa device is used as the receiver. For each experiment, we transmit at least 1000 packets with random payloads, and repeat each experiment three times. Unless differently specified, the coding rate is set to 1.

### 5.1 Reliability

We start by evaluating the reliability of EMU's three major functions (chirp emulation, snipping, and multiplexing) over multiple combinations of physical layer settings (i.e., SF, LDRO, gap duration).

**Experimental setup.** We place a transmitter (running EMU) and a receiver node (running classical LoRa) in very close proximity ($\approx$10 cm) to evaluate the reliability of chirp emulation and snipping while ensuring a sufficient link budget[7]. We investigate ten combinations of physical layer settings (i.e., SF 7 to 12 with default LDRO configuration as well as SF 7 to 10 with the LDRO enabled manually): for each combination, we study the reliability of chirp snipping when using different gap durations, i.e., snipping from 5% to 70% of a symbol in steps of 5%. We always snip the chips at the end of a chirp, since this leads to the most reliable configuration, as shown in Sec. 3.2. Since the length of a packet affects reliability as well, i.e., the longer a packet, the lower the likelihood of a successful reception [2], we send packets with a length of 255 bytes, so to have a worst-case scenario. When evaluating the performance of chirp multiplexing, we enable the LDRO for all SFs and snip 50% of

[7]Please note that in this section we use a table setup only to simplify experimentation. As shown in Sec. 6, EMU also works in a network deployed over a campus.

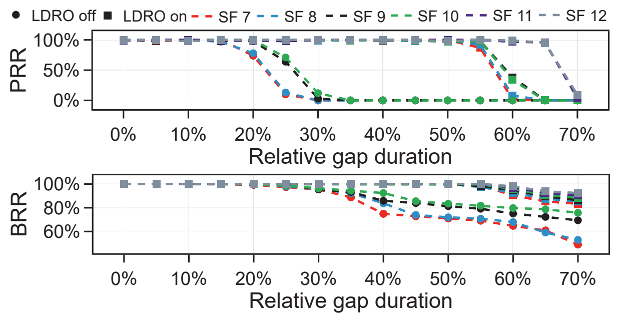


**Figure 9: Reliability of chirp emulation and snipping.** When emulating (gap duration of 0%), the PRR and the BRR are 100% regardless of the employed SF and LDRO configuration. When snipping, enabling the LDRO allows to snip more than 50% of a chirp with any SF, without significant impact on PRR or BRR.

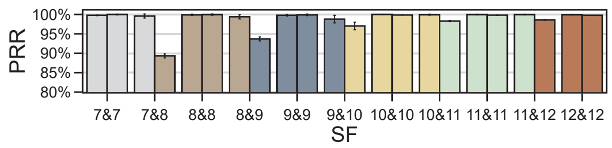


**Figure 10: Reliability of chirp multiplexing.** When concurrently sending two packets with the same SF, the reliability is ≈100%. When using different SF, the PRR is lower (but still above 90%).

| **SF** | 7 | 8 | 9 | 10 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|
| **LDRO** | D | D | D | D | E | E | E | E | E | E |
| **Max. gap (%)** | 15 | 15 | 20 | 20 | 55 | 55 | 55 | 55 | 65 | 65 |

**Table 1: Maximum gap duration that can be used when snipping chirps before the PRR starts to decline.** Acronyms E and D refer to the enabling and disabling of the LDRO, respectively.

the chirp, regardless of the employed SF. We make use of a second receiver, also equipped with an SX1276 radio, to overhear communications on a second channel, and compute the corresponding PRR.

**Results.** Fig. 9 shows the PRR (top) and BRR (bottom) as a function of the relative gap duration, respectively. In absence of gaps (i.e., when performing a chirp emulation), the PRR is ≈100% regardless of the employed SF and LDRO settings; that is, a commercial LoRa radio can successfully decode the packets emulated by EMU. This confirms, among others, that we correctly understood how the LDRO works, and that the description provided in Sec. 2.3 is correct. When the gap duration increases, the PRR remains close to 100%, but starts to decline when using a SF of 7 and 8 as soon as more than 15% of a chirp is snipped. Similarly, the PRR starts to decline when using a SF of 9 and 10 when more than 25% of a chirp is snipped. Note that these four configurations have the LDRO disabled by default: when enabling this feature, the gap duration can be increased up to 55% before the PRR is affected. Similarly, when making use of SF 11 and 12, which enable the LDRO by default, a PRR of ≈100% can be sustained when snipping a chirp by up to 65%. This observation is important: in order to enable chirp multiplexing (i.e., to transmit two chirps simultaneously), we need to be able to snip at least 50% of a chirp: therefore, to support chirp multiplexing, we enable the LDRO with all spreading factors. Tab. 1 summarizes the maximum gap duration before the performance (PRR) starts to deteriorate.

Fig. 10 depicts the reliability of chirp multiplexing for different configurations. When the same SF is used, all the multiplexed packets can be decoded successfully by both receivers (PRR ≈ 100%). When using different (adjacent) SFs, the reliability of the packets sent with the higher SF is slightly below 100% (albeit still greater than 90%). This is likely due to the fact that chirp multiplexing also involves the transmission of the preamble, sync word, and down-chirps: as discussed in Sec. 3.3: these are more prone to decoding errors than the header and payload [28]. Hence, having only a quarter of the chirp sent consecutively when using a higher SF (rather than the half of a chirp when using the same SF) results in a slightly lower reliability, as illustrated in Fig. 10.

## 5.2 Energy Consumption

**Experimental setup.** We measure the current drawn by the transmitter using a Tektronix DMM7510 digital multimeter. We keep the TX power constant (5 dBm), and derive the energy consumption of EMU when performing chirp emulation, snipping, and multiplexing, comparing it to that of traditional LoRa. When snipping, we employ the maximum gap duration specified in Tab. 1 and LoRaWAN's default LDRO configuration. When multiplexing, the LDRO is enabled for all SFs and we snip 50% of the chirp. We use both short (10 bytes) and long (255 bytes) payloads. Unless differently specified, we always account for the energy costs of both radio and MCU.

**Results.** Fig. 11 shows how both chirp snipping and multiplexing allow to significantly reduce energy consumption. Specifically, the figure shows how much energy is saved *per transmitted byte* when EMU snips and multiplexes chirps, using traditional LoRa with the same PHY settings as a baseline. EMU's pure chirp emulation results in a slightly higher energy consumption (roughly between 4% and 0.7% higher than traditional LoRa): this is due to the intrinsic limitations of performing the data encoding in software rather than in hardware, as well as to the use of the CLKOUT pin to output a reference clock as timer source (see Sec. 4.3). When performing chirp snipping, the longer the gap duration, the more energy can be saved: for this reason, the higher the SF, the higher the energy savings, which can be as high as 30% when using SF 11 and 12. In contrast, when using lower SFs, the benefits of snipping are only as high as 10% when sending large packets. For shorter packets and low SFs, the energy is comparable or slightly lower than that of traditional LoRa: the reason for this is that snipping only occurs *after* the transmission of preamble, sync word and down-chirps. Thus, the number of transmitted symbols is lower with shorter payloads, which limits the energy savings achieved by EMU. When performing chirp multiplexing, the concurrent transmission of two packets allows much higher energy savings. Our results show that one can save between 28% and 36% of energy when using SFs from 7 to 10, and as much as 49% for SF 11 and 12.

## 5.3 Receiver Sensitivity

We quantify in this section the loss in receiver sensitivity of EMU when emulating, snipping, and multiplexing chirps as a function of the employed SF, and compare it to that of traditional LoRa.

**Experimental setup.** To evaluate the receiver sensitivity, we connect transmitter and receiver nodes with an SMA cable. We cascade two 90 dB adjustable attenuators (Mini-Circuits RCDAT-8000-90), which allows us to decrease the signal power between -100 and

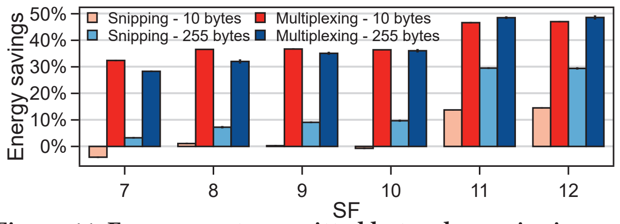


**Figure 11: Energy per transmitted byte when snipping and multiplexing chirps.** Both functionalities of EMU allow to save significant energy compared to traditional LoRa communications.

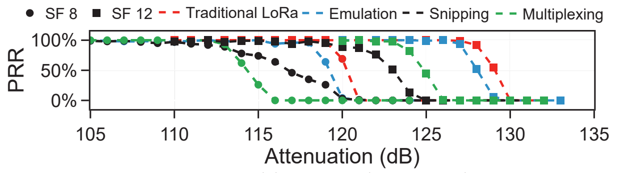


**Figure 12: PRR sustained by EMU when emulating, snipping, and multiplexing chirps compared to that of traditional LoRa.** The receiver sensitivity decreases by ≈2, 8, and 6 dB when running EMU in emulating, snipping, and multiplexing mode, respectively.

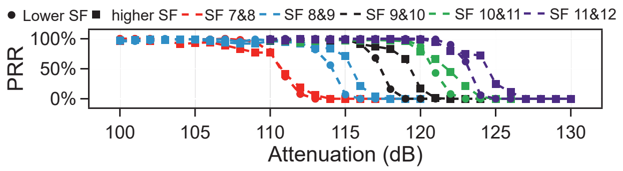


**Figure 13: PRR sustained by EMU when multiplexing chirps with different SFs.** The decrease in PRR is slightly higher for packets sent with a higher SF, in line with the results in Sec. 5.1.

-135 dBm in 1 dB steps. We study how the PRR decreases as a function of the attenuation for different SFs and the corresponding LDRO configurations. As in Sec. 5.1, we use large LoRa packets (255 bytes of random payload), to account for the worst case. When snipping chirps, we generate gaps as listed in Tab. 1. When multiplexing chirps, we compute the PRR on both channels and report its average (but note that the PRR is similar across both receivers).

**Results.** Fig. 12 shows the PRR of traditional LoRa and that of EMU when emulating, snipping, and multiplexing chirps as a function of the introduced attenuation. We show the PRR when adopting a low spreading factor (SF=8, circle) as well as a high one (SF=12, square). As expected, the receiver sensitivity is higher when making use of traditional LoRa devices, as they can generate packets directly in hardware. Compared to traditional LoRa, EMU's chirp emulation reduces the receiver sensitivity by roughly 1–2 dB, regardless of the employed spreading factor. When performing chirp snipping and multiplexing, the loss in sensitivity compared to traditional LoRa increases to approximately 8 and 6 dB[8], respectively. Such an increase in sensitivity loss is expected because, as mentioned in Sec. 3, the chirps are incomplete and sent with a lower frequency granularity. Still, it is still possible to reach a receiver sensitivity lower than -120 dBm, which is one of the main features of LoRa technology [59]. Fig. 13 shows the PRR as a function of the signal attenuation when EMU performs chirp multiplexing of packets sent

[8]We notice that the PRR declines more steeply when performing chirp multiplexing: this is why the PRR of snipping will be higher than that of multiplexing in Sec. 6.2.

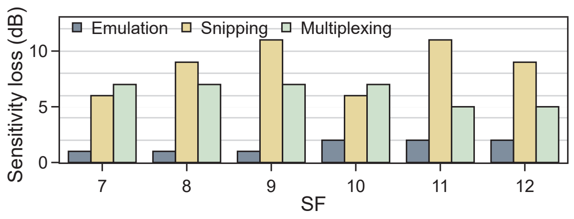


**Figure 14: Comparison of the sensitivity loss when using EMU in emulation, snipping, and multiplexing mode to that of traditional LoRa.** The values are computed based on the receiver sensitivity that allows to achieve a PRR > 95%.

with different SF. In line with the results presented in Sec. 5.1, the decrease in PRR is stronger for packets sent with a higher SF, and the loss in sensitivity is similar to the one shown in Fig. 12. We finally summarize the sensitivity loss when using EMU compared to that of traditional LoRa in Fig. 14. The values displayed in the histogram are derived by picking the receiver sensitivity that allows to achieve a PRR above 95%. As discussed in Sec. 3.2, a sensitivity below -120 dBm still suffices to transmit data over large distances (up to 2700 m), which is sufficient for most LoRa-based applications [35, 83]. Thus, our experimental results confirm that the use of chirp snipping can find applicability in many real-world deployments.

## 6 USING EMU WITHIN A LORAWAN

In this section, we run EMU as transmitter on both LoRa (SX1276) and non-LoRa platforms (TI CC1125), and establish a communication to a LoRaWAN gateway (RAK7243 with ChirpStack[9]), showing that:

- The LoRaWAN uplink throughput can be improved significantly thanks to the use of chirp multiplexing (Sec. 6.1);
- EMU's chirp emulation, snipping, and multiplexing work well also in a real deployment across large distances (Sec. 6.2);
- Even non-LoRa nodes can directly send information to a LoRaWAN gateway using EMU (Sec. 6.2).
- The use of EMU's chirp emulation allows non-LoRa nodes to achieve a longer range (Sec. 6.2).

### 6.1 Throughput Improvements

We quantify the throughput improvements introduced by EMU's chirp multiplexing and compare it with that of traditional LoRa.

**Experimental setup.** We let an SX1276 node run EMU while working as an Activation By Personalization (ABP) LoRaWAN device [71]. With ABP, the information for joining a LoRaWAN (i.e., a fixed device address and session key) provided by the gateway is hard-coded into the device, so that the latter can immediately join the LoRaWAN and exchange data with the gateway. We also configure EMU to comply with the regulated duty cycle limitations (1%) on each channel[10]. There are eight fixed channels (from 486.3 to 487.7 MHz with a channel spacing of 200 kHz) overheard by the gateway. When performing chirp multiplexing, we send two packets using the same SF, and remap the channel hopping mechanism used by traditional LoRa nodes since the two channels used when multiplexing have

[9]https://github.com/brocaar/chirpstack-docker, commit: 2c9ee5e.

[10]This results in packets being sent every 4.6, 8.2, 8.5, 8.7, 19.5, and 35 s for SF 7 to 12, respectively, when using traditional LoRa and EMU with chirp emulation. When using chirp multiplexing, packets are sent every 6.3, 10.6, 10.5, 10.8, 19.5, and 35 s for SF 7 to 12. The payload length is 222, 222, 115, 51, 51, and 51 bytes for SF 7 to 12, respectively.

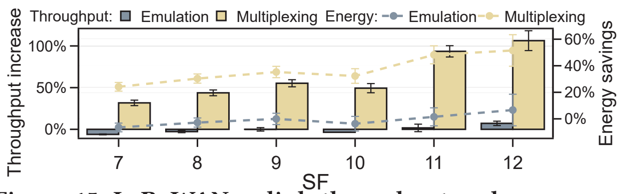


**Figure 15: LoRaWAN uplink throughput and energy consumption per received byte when using chirp emulation and multiplexing.** The throughput can be doubled and up to 51% energy per received byte can be saved thanks to EMU.

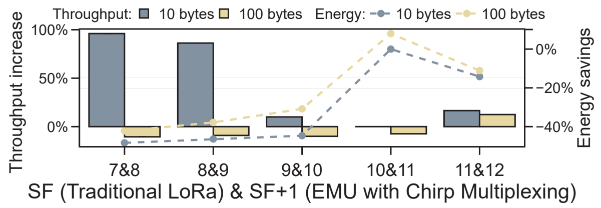


**Figure 16: Performance when multiplexing chirps using EMU with a higher spreading factor (SF+1) than that used by traditional LoRa (SF) in a LoRaWAN context.** When transmitting short payloads, EMU still achieves a higher throughput.

to be > 400 kHz (see Sec. 3.3). We use the maximum payload length allowed in LoRaWAN [41], and record the number of packets received by the gateway over 10 minutes runs. As for Sec. 5.1, when evaluating the performance of chirp multiplexing, we enable the LDRO for all SFs and snip 50% of the chirp, regardless of the employed SF. Accordingly, on the gateway, we enable the LDRO by setting `PPM_OFFSET` in `RegMisc_cfg2` and `MBWSSF_PPM_OFFSET` in `RegMbwssf_misc_cfg2`[11]. After sending packets to the LoRa gateway using EMU, the transmitting node switches back to receiving mode to get the ACK packets sent by the gateway. Note that the gateway will receive both packets at the same time, but will only respond with an ACK to one of them, due to a bug in the implementation of the LoRa network server. In these experiments, the gateway is located in the same room of the SX1276 transmitter, at a distance of ≈10 m. Besides quantifying the improvements in uplink throughput, we also monitor energy consumption using a software-based energy estimation approach resembling Energest [19].

**Results.** The bars in Fig. 15 show the difference in uplink throughput when using EMU with chirp multiplexing compared to traditional LoRa. The throughput of EMU when performing chirp multiplexing is significantly higher regardless of the employed SF. For SF 11 and 12, we observe the highest increase in throughput, which is essentially doubled compared to that of traditional LoRa. This is because the latter does not use the LDRO option for SF 7–10, which results in a faster data transmission. The lines in Fig. 15 show the energy per received byte, which is even 59% lower than when using traditional LoRa, in line with the findings presented in Fig. 11. The difference in throughput and energy of EMU when performing chirp emulation are also plotted as a reference, and are close to zero.

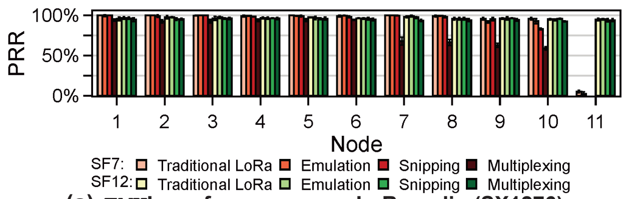
**Discussion.** The aforementioned results show the benefits introduced by EMU compared to a system running LoRa using the same PHY settings. In principle, from a pure PHY perspective (i.e., without accounting for the use of LoRaWAN), a LoRa device could make use of a lower SF to obtain *equivalent of higher benefits* than those introduced by EMU's snipping and multiplexing. However, when using LoRaWAN and small payloads, the throughput of traditional LoRa with a lower SF is not necessarily better than that of EMU with chirp multiplexing, as shown in Fig. 16. The better performance of EMU with short payloads is due to the mandatory receive window (typically 2 s) after transmitting a packet [41, 56]: this introduces a delay during which packets cannot be sent (EMU has an advantage, as it is able to send two packets simultaneously). With high spreading factors and lower data rates, the delays due to the radio duty cycle regulations (which are much longer than the receive window) are the dominating factor, diminishing EMU's benefits. Independently of these results, adopting a lower SF assumes that a node has the ability to freely choose the SF to be used for communication. While this is typically the case with LoRaWAN gateways and static end-nodes, the same does not hold true for P2P communication across LoRa end-nodes (where the SF is fixed and cannot be chosen arbitrarily for each individual packet transmission) and in the presence of mobile end-nodes communicating to a gateway (where blind ADR is recommended by LoRaWAN [60]). For such scenarios, the ability to snip and multiplex chirps sent with a given SF with EMU can be an asset to increase throughput and reduce energy consumption.

## 6.2 Deploying EMU in a University Campus

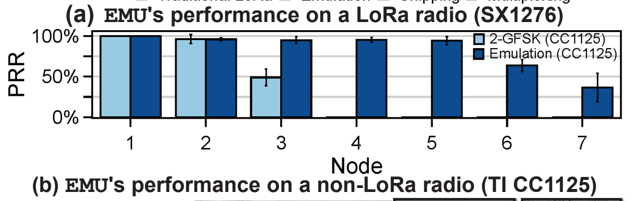
We conclude our experimental evaluation by deploying several LoRa (SX1276) and non-LoRa (TI CC1125) nodes running EMU across a University campus, and evaluate their performance by connecting them to a LoRaWAN gateway keeping track of the received packets from each node. For the SX1276 nodes, we evaluate the performance of EMU in chirp emulation, snipping, and multiplexing mode, against that of traditional LoRa. For the TI CC1125 nodes, we evaluate the performance of EMU when communicating to the LoRaWAN gateway using chirp emulation, and compare it to the performance obtained by the device using its native 2-GFSK modulation to communicate with another TI CC1125 device acting as receiver (we place this device in proximity of the LoRaWAN gateway to ensure a fair comparison). Fig. 17(c) shows the position of the LoRaWAN gateway and of all other nodes used in our evaluation.

**EMU's performance on a LoRa radio.** We let the SX1276 devices running EMU (marked as green circles in Fig. 17(c)) send 100 packets to the LoRaWAN gateway using chirp emulation, snipping, and multiplexing, for different SFs (7 and 12) as well as a fixed transmission power (14 dBm). We repeat each run three times. When running EMU in chirp snipping mode, we use a gap of 15% with LDRO disabled for SF=7, and a 65% gap with LDRO enabled for SF=12. When running EMU in chirp multiplexing mode, we use a gap of 50% and enable LDRO for both SF 7 and 12. Fig. 17(a) shows the PRR measured at the LoRaWAN gateway broken down per individual node and EMU mode. We can see that the LoRaWAN gateway receives most of the packets transmitted by EMU correctly, also when nodes are relatively far (nodes 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 and 11 are placed 150, 300, 400, 350, 350, 350, 500, 900, 1100, 1350 and 1500 m away from the gateway, respectively). Please note that although the PRR of EMU when using chirp multiplexing is approximately 100%, the actual number of packets sent is double compared to the one

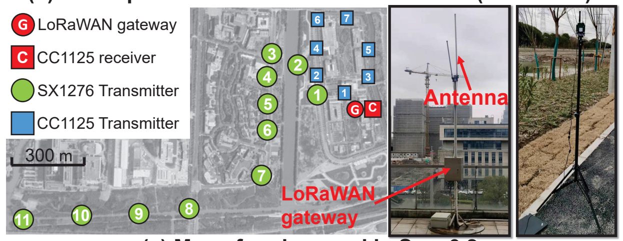
[11]Please note that enabling the LDRO when using a lower SF than 11 may lead to potential compatibility issues with existing LoRaWAN gateways. However, note that this was not the case for the gateway used in our experiments (RAK7243 with ChirpStack).



**(a) EMU's performance on a LoRa radio (SX1276)**



**(b) EMU's performance on a non-LoRa radio (TI CC1125)**



**(c) Map of nodes used in Sec. 6.2**

**Figure 17: Performance of EMU in an outdoor deployment (c) using both LoRa and non-LoRa radios.** EMU sustains a high PRR also at large distances (a). Non-LoRa radios using EMU can transmit data to LoRaWAN gateways and increase their range (b).

sent by traditional LoRa, as two packets are sent concurrently on two separate channels. The PRR sustained by EMU with SF=7 starts to decline after 1.3 km, but the gateways can still receive packets sent with SF=12 at higher distances (e.g., node 11).

**EMU's performance on a non-LoRa radio.** We let the TI CC1125 nodes running EMU (marked as blue circles in Fig. 17(c)) send 100 packets to the LoRaWAN gateway using chirp emulation, SF=12, and a fixed transmission power (0 dBm). The same nodes also send packets using their native 2-GFSK modulation to a receiver placed nearby the gateway using the same transmission power. We repeat each run three times. Fig. 17(b) shows that EMU's chirp emulation allows the non-LoRa devices to successfully communicate with the LoRaWAN gateway and to increase their communication range. In fact, nodes 4–7 cannot communicate with the CC1125 receiver when using their native 2-GFSK modulation, but can do so using EMU. These results hence confirm that the use of EMU is beneficial even for non-LoRa transceivers, which can increase their communication range and communicate directly to a LoRaWAN gateway, thereby enabling new use cases and expanding the applicability of LoRa.

## 7 RELATED WORK

We analyse related work with respect to (i) the body of literature increasing the energy efficiency and throughput of LoRa-based solutions, as well as (ii) solutions enabling a cross-technology communication (CTC) between LoRa devices and other technologies.

**Improvement of LoRa-based solutions.** The architecture of LoRaWANs is a star topology, where end-devices send data directly to LoRaWAN gateways [41] making use of protocols based on ALOHA [41]. Nodes can transmit in parallel by utilizing the orthogonality between combinations of LoRa parameters in terms of SF, frequency and BW [7, 84]. However, this still results in a high number of collisions [47], especially when nodes are placed densely [1, 5], which leads to a higher packet loss, latency, and energy expenditure. Therefore, researchers have proposed several techniques to avoid such collisions and mitigate their impact. These include lightweight scheduling algorithms [50], TDMA-based mechanisms [76], optimal resource allocation schemes [12, 21, 25] and novel carrier sensing approaches [24, 36]. Moreover, the use of directional antennae [77] also allows to mitigate the detrimental impact of packet collisions. Researchers have also derived solutions to disentangle collided packets at the gateway [20, 31, 74, 75, 78, 81] and to receive more information from received packets [42] or waveforms [18], thereby improving goodput and reducing the sending attempts of end devices. A few works such as Chime [23] have also focused on mitigating the effects of multipath by means of an improved frequency selection. In contrast to all these works, EMU improves the efficiency and throughput by modifying the PHY (i.e., by snipping and multiplexing chirps) on the transmitting device.

**Cross-technology communication with LoRa devices.** CTC enables devices with different radio technologies to directly communicate without the need of multi-radio gateways [32, 37]. Most of the work on CTC has focused on Wi-Fi, BLE, and IEEE 802.15.4 (ZigBee) devices operating in the 2.4 GHz ISM band [15, 29, 30, 82]. With the emergence of LoRa devices operating at 2.4 GHz [58], a few works have also enabled CTC between LoRa and other technologies sharing these frequencies. For example, XFi [40] enables Wi-Fi receivers to receive simultaneously eight streams of LoRa packets; BLE2LoRa [38] allows BLE devices to communicate with LoRa devices; whereas Wi-Lo [26] allows a Wi-Fi device to send LoRa-compliant packets. Whilst BLE2LoRa [38] and Wi-Lo [26] also emulate a CSS signal, they only embed LoRa symbols within a Wi-Fi or BLE packet. EMU, instead, constructs each individual chirp chip-by-chip, which results in a more fine-grained emulation. c-Chirp [80] builds a chirp-based CTC channel between ZigBee and Wi-Fi, but without targeting LoRa devices. Finally, LoRaBee [61] achieves CTC from LoRa to Zigbee devices operating in the sub-GHz band, whereas EMU allows non-LoRa devices to communicate to off-the-shelf LoRa receivers and LoRaWAN gateways.

## 8 CONCLUSIONS

We have presented EMU, a framework that enables the emulation, snipping, and multiplexing of LoRa chirps on commercial IoT devices equipped with low-power sub-GHz transceivers. By emulating chirps with a (G)FSK/OOK modulator, EMU generates LoRa packets that can be decoded by off-the-shelf LoRa receivers and LoRaWAN gateways. Chirp snipping, i.e., artificially removing a sequence of chips and placing the radio in low-power mode, allows to reduce the energy consumption while retaining a reliable communication. Chirp multiplexing exploits the gaps introduced by chirp snipping to transmit portions of another chirp on a separate channel, thereby enabling the concurrent transmission of two LoRa packets at once, which allows to increase throughput. We strongly believe that EMU's modular framework and open-source availability will enable new use cases and expand the applicability of LoRa technology.

## REFERENCES

[1] F. Adelantado et al. 2017. Understanding the Limits of LoRaWAN. *IEEE Commun. Mag.* 55, 9 (2017).

[2] T. Attia et al. 2019. Experimental Characterization of Packet Reception Rate in LoRaWAN. In *Proc. of the 4th CoRes Conf.*

[3] A. Augustin et al. 2016. A Study of LoRa: Long Range & Low Power Networks for the Internet of Things. *Sensors* 16, 9 (2016).

[4] W. Ayoub et al. 2020. Technology Selection for IoT-Based Smart Transportation Systems. In *Vehicular Ad-hoc Networks for Smart Cities*.

[5] D. Bankov et al. 2016. On the Limits of LoRaWAN Channel Access. In *Proc. of the 3rd EnT Conf.*

[6] P. J. Basford et al. 2020. LoRaWAN for Smart City IoT Deployments: A Long Term Evaluation. *Sensors* 20, 3 (2020).

[7] M. Bor et al. 2016. Do LoRa Low-Power Wide-Area Networks Scale?. In *Proc. of the 19th MSWiM Conf.*

[8] M. Bor et al. 2016. LoRa for the Internet of Things. In *Proc. of the 1st MadCom Workshop*.

[9] M. Bor et al. 2017. LoRa Transmission Parameter Selection. In *Proc. of the 13th DCOSS Conf.*

[10] T. Bouguera et al. 2018. Energy Consumption Model for Sensor Nodes Based on LoRa and LoRaWAN. *Sensors* 18, 7 (2018).

[11] C. Bouras et al. 2021. Energy Efficient Mechanism for LoRa Networks. *Internet of Things* 13 (2021).

[12] C. Caillouet et al. 2019. Optimal SF Allocation in LoRaWAN Considering Physical Capture and Imperfect Orthogonality. In *Proc. of the 38th GLOBECOM Conf.*

[13] M. Cattani et al. 2017. Adige: An Efficient Smart Water Network based on Long-Range Wireless Technology. In *Proc. of the 3rd CySWATER Workshop*.

[14] M. Cattani et al. 2017. An Experimental Evaluation of the Reliability of LoRa Long-Range Low-Power Wireless Communication. *JSAN* 6, 2 (2017).

[15] K. Chebrolu et al. 2009. Esense: Communication through Energy Sensing. In *Proc. of the 15th MobiCom Conf.*

[16] Y. Cheng et al. 2018. Secure Smart Metering based on LoRa Technology. In *Proc. of the 4th ISBA Conf.*

[17] S. Demetri et al. 2019. Automated Estimation of Link Quality for LoRa: A Remote Sensing Approach. In *Proc. of the 18th IPSN Conf.*

[18] A. Dongare et al. 2018. Charm: Exploiting Geographical Diversity through Coherent Combining in LP-WANs. In *Proc. of the 17th IPSN Conf.*

[19] A. Dunkels et al. 2007. Software-based On-line Energy Estimation for Sensor Nodes. In *Proc. of the 4th EmNetS Workshop*.

[20] R. Eletreby et al. 2017. Empowering low-power wide area networks in urban settings. In *Proc. of the 31st SIGCOMM Conf.*

[21] A. Farhad et al. 2020. Resource Allocation to Massive Internet of Things in LoRaWANs. *Sensors* 20, 9 (2020).

[22] B. Foubert et al. 2020. Long-Range Wireless Radio Technologies: A Survey. *Future Internet* 12, 1 (2020).

[23] A. Gadre et al. 2020. Frequency Configuration for Low-Power Wide-Area Networks in a Heartbeat. In *Proc. of the 17th NSDI Conf.*

[24] A. Gamage et al. 2020. LMAC: Efficient Carrier-Sense Multiple Access for LoRa. In *Proc. of the 26th MobiCom Conf.*

[25] W. Gao et al. 2020. AdapLoRa: Resource Adaptation for Maximizing Network Lifetime in LoRa networks. In *Proc. of the 28th ICNP Conf.*

[26] P. Gawlowicz et al. 2021. Wi-Lo: Emulating LoRa using COTS WiFi. *CORR – arXiv preprint 2105.04998* (2021).

[27] R. Ghanaatian et al. 2019. Lora Digital Receiver Analysis and Implementation. In *Proc. of the 44th ICASSP Conf.*

[28] J. Haxhibeqiri et al. 2018. Sub-Gigahertz Inter-Technology Interference. How Harmful is it for LoRa?. In *Proc. of the 4th ISC2 Conf.*

[29] R. Hofmann et al. 2019. X-Burst: Enabling Multi-Platform Cross-Technology Communication between Constrained IoT Devices. In *Proc. of the SECON Conf.*

[30] R. Hofmann et al. 2021. SERVOUS: Cross-Technology Neighbour Discovery and Rendezvous for Low-Power Wireless Devices. In *Proc. of the 18th EWSN Conf.*

[31] B. Hu et al. 2020. SCLoRa: Leveraging Multi-Dimensionality in Decoding Collided LoRa Transmissions. In *Proc. of the 28th ICNP Conf.*

[32] W. Jiang et al. 2017. BlueBee: A 10,000x Faster Cross-Technology Communication via PHY Emulation. In *Proc. of the 15th ACM SenSys Conf.*

[33] jkadbear. 2021. LoRaPHY – A complete MATLAB implementation of LoRa PHY. [Online] https://github.com/jkadbear/LoRaPHY – Last access: 2021-10-29.

[34] B. Kempke et al. 2016. SurePoint: Exploiting UWB Flooding and Diversity to Provide Robust, Scalable, High-Fidelity Indoor Localization. In *Proc. of the 14th ACM SenSys Conf.*

[35] R. K. Kodali et al. 2018. Smart Farm Monitoring Using LoRa Enabled IoT. In *Proc. of the 2nd ICGCIoT Conf.*

[36] N. Kouvelas et al. 2020. P-CARMA: Politely Scaling LoRaWAN. In *Proc. of the 17th EWSN Conf.*

[37] Z. Li et al. 2017. WEBee: Physical-Layer Cross-Technology Communication via Emulation. In *Proc. of the 23rd MobiCom Conf.*

[38] Z. Li et al. 2020. BLE2LoRa: Cross-Technology Communication from Bluetooth to LoRa via Chirp Emulation. In *Proc. of the 17th SECON Conf.*

[39] C.-H. Liao et al. 2016. Revisiting the So-Called Constructive Interference in Concurrent Transmission. In *Proc. of the 41st LCN Conf.* IEEE.

[40] R. Liu et al. 2020. XFi: Cross-technology IoT Data Collection via Commodity WiFi. In *Proc. of the 28th ICNP Conf.*

[41] LoRa Alliance. 2017. LoRaWAN Specification, v1.1. [Online] https://bit.ly/2F0QbQM – Last access: 2021-10-29.

[42] P. J. Marcelis et al. 2017. DaRe: Data Recovery through Application Layer Coding for LoRaWAN. In *Proc. of the 2nd IoTDI Conf.*

[43] K. Mekki et al. 2019. A Comparative Study of LPWAN Technologies for Large-Scale IoT Deployment. *ICT Express* 5, 1 (2019).

[44] N. Naik et al. 2018. LPWAN Technologies for IoT Systems: Choice Between Ultra Narrow Band and Spread Spectrum. In *Proc. of the 4th ISSE Conf.*

[45] M. N. Ochoa et al. 2017. Evaluating LoRa Energy Efficiency for Adaptive Networks: From Star to Mesh Topologies. In *Proc. of the 13th WiMob Conf.*

[46] J. Petäjäjärvi et al. 2015. On the Coverage of LPWANs: Range Evaluation and Channel Attenuation Model for LoRa Technology. In *Proc. of the* $14^{th}$ *ITST Conf.*

[47] A. Rahmadhani et al. 2018. When LoRaWAN Frames Collide. In *Proc. of the 12th WiNTECH Conf.*

[48] U. Raza et al. 2017. Low Power Wide Area Networks: An Overview. *IEEE Commun. Surv. Tutor.* 19, 2 (2017).

[49] Red Oak Farm. [n.d.]. Learning about Emu Feathers. [Online] http://www.redoakfarm.com/learning_about_feathers.htm – Last access: 2021-10-29.

[50] B. Reynders et al. 2018. Improving Reliability and Scalability of LoRaWANs Through Lightweight Scheduling. *IoT-J* 5, 3 (2018).

[51] P. Robyns et al. 2018. A Multi-Channel Software Decoder for the LoRa Modulation Scheme. In *Proc. of the 3rd IoTBDS Conf.*

[52] M. Saelens et al. 2019. Impact of EU Duty Cycle and Transmission Power Limitations for Sub-GHz LPWAN SRDs: An Overview and Future Challenges. *J. Wirel. Commun. Netw.* (2019).

[53] M. Sandell et al. 2019. Application Layer Coding for IoT: Benefits, Limitations, and Implementation Aspects. *IEEE Syst J.* 13, 1 (2019).

[54] Semtech. 2015. AN1200. 22 LoRa Modulation Basics. [Online] https://sforce.co/30OFbAV – Last access: 2021-10-29.

[55] Semtech. 2018. Application Note: Recommendations for Best Performance. [Online] https://bit.ly/3G9h0Nu – Last access: 2021-10-29.

[56] Semtech. 2019. An In-depth Look at LoRaWAN Class A Devices. [Online] https://bit.ly/3H3oalN – Last access: 2021-10-29.

[57] Semtech. 2019. SX1268 Datasheet. [Online] https://sforce.co/3pnhGJg – Last access: 2021-10-29.

[58] Semtech. 2020. Long Range, Low Power, 2.4 GHz Transceiver with Ranging Capability. [Online] https://bit.ly/2F0QbQM – Last access: 2021-10-29.

[59] Semtech. 2020. SX1276-7-8-9 Datasheet. [Online] https://sforce.co/3lWhC11 – Last access: 2021-10-29.

[60] Semtech's LoRa Developer Portal. 2021. LoRa Device Mobility: An Introduction to Blind ADR. [Online] https://bit.ly/3sUO5Hc – Last access: 2021-10-29.

[61] J. Shi et al. 2019. LoRaBee: Cross-Technology Communication from LoRa to ZigBee via Payload Encoding. In *Proc. of the 27th ICNP Conf.*

[62] N. Silva et al. 2019. Low-Cost IoT LoRa Solutions for Precision Agriculture Monitoring Practices. In *Proc. of the* $19^{th}$ *EPIA Conf.*

[63] J. P. S. Sundaram et al. 2020. A Survey on LoRa Networking: Research Problems, Current Solutions, and Open Issues. *IEEE Commun. Surv. Tutor.* 22, 1 (2020).

[64] F. Sutton et al. 2015. Bolt: A Stateful Processor Interconnect. In *Proc. of the 13th ACM SenSys Conf.*

[65] F. Sutton et al. 2017. The Design of a Responsive and Energy-efficient Event-triggered Wireless Sensing System. In *Proc. of the 14th EWSN Conf.*

[66] J. Tapparel. 2019. *Complete Reverse Engineering of LoRa PHY*. Technical Report. EPFL, Lausanne.

[67] J. Tapparel et al. 2020. An Open-Source LoRa Physical Layer Prototype on GNU Radio. In *Proc. of the 21st SPAWC Conf.* 1–5.

[68] L. Tessaro et al. 2018. LoRa Performance in Short Range Industrial Applications. In *Proc. of the 23rd SPEEDAM Symp.*

[69] Texas Instruments. 2009. CC1100 Low-Power Sub- 1 GHz RF Transceiver. [Online] https://bit.ly/3G1yYSh – Last access: 2021-10-29.

[70] Texas Instruments. 2014. CC1125 Ultra-High Performance RF Narrowband Transceiver. [Online] https://bit.ly/2ZavXhC – Last access: 2021-10-29.

[71] The Things Network. 2021. ABP vs OTAA. [Online] https://www.thethingsindustries.com/docs/devices/abp-vs-otaa/ – Last access: 2021-10-29.

[72] The Things Network. 2021. Regional Parameters. [Online] https://bit.ly/3JFkOas – Last access: 2021-10-29.

[73] P. Tian et al. 2021. ChirpBox: An Infrastructure-Less LoRa Testbed. In *Proc. of the 18th EWSN Conf.*

[74] S. Tong et al. 2020. CoLoRa: Enabling Multi-Packet Reception in LoRa. In *Proc. of the 39th INFOCOM Conf.*

[75] S. Tong et al. 2020. Combating Packet Collisions Using Non-Stationary Signal Scaling in LPWANs. In *Proc. of the 18th MobiSys Conf.*

[76] R. Trüb et al. 2018. Increasing Throughput and Efficiency of LoRaWAN Class A. In *Proc. of the 12th UBICOMM Conf.*
[77] T. Voigt et al. 2017. Mitigating Inter-Network Interference in LoRa Networks. In *Proc. of the 14th EWSN Conf.*
[78] X. Wang et al. 2019. mLoRa: A Multi-Packet Reception Protocol in LoRa networks. In *Proc. of the 27th ICNP Conf.*
[79] Z. Wang et al. 2020. Online Concurrent Transmissions at LoRa Gateway. In *Proc. of the IEEE INFOCOM Conf.*
[80] D. Xia et al. 2020. c-Chirp: Towards symmetric cross-technology communication over asymmetric channels. In *Proc. of the 17th SECON Conf.*
[81] X. Xia et al. 2019. FTrack: Parallel Decoding for LoRa Transmissions. In *Proc. of the 17th ACM SenSys Conf.*
[82] Z. Yu et al. 2018. Crocs: Cross-Technology Clock Synchronization for WiFi and ZigBee. In *Proc. of the 15th EWSN Conf.*
[83] W. Zhao et al. 2017. Design and Implementation of Smart Irrigation System Based on LoRa. In *Proc. of the 1st LPWA4IoT Workshop.*
[84] D. Zorbas et al. 2018. Improving LoRa Network Capacity Using Multiple Spreading Factor Configurations. In *Proc. of the 25th ICT Conf.*