SDR-LoRa: Dissecting and Implementing LoRa on Software-Defined Radios to Advance Experimental IoT Research

Fabio Busacca*, Stefano Mangione*, Ilenia Tinnirello*§, Sergio Palazzo§, and Francesco Restuccia†
* Department of Engineering, University of Palermo, Italy
§Department of Electrical, Electronics and Computer Engineering (DIEEI), University of Catania, Italy
† Institute for the Wireless Internet of Things, Northeastern University, Boston, USA

ABSTRACT

In this paper, we present SDR-LoRa, a full-fledged SDR implementation of a LoRa transmitter and receiver. First, we reverse-engineer the LoRa physical layer (PHY) functionalities, including the procedures of packet modulation, demodulation, and preamble detection. Based on this analysis, we develop the first Software Defined Radio (SDR) implementation of the LoRa PHY. Furthermore, we integrate LoRa with an Automatic Repeat Request (ARQ) error detection protocol. SDR-LoRa has been validated on (i) the Colosseum wireless channel emulator; and (ii) a real testbed with USRP radios and commercial-off-the-shelf (COTS) devices. Our experimental results demonstrate that the performance of SDR-LoRa is in line with commercial LoRa systems. We pledge to share the entirety of the SDR-LoRa code.

CCS CONCEPTS

• **Networks** → **Network experimentation**; **Network performance analysis**; **Network measurement**; **Link-layer protocols**; **Wide area networks**; • **Hardware** → *Wireless devices*; Sensor devices and platforms; • **Computer systems organization** → *Sensor networks*; *Reliability*.

KEYWORDS

LoRa, Software Defined Radio, LPWAN, IoT

ACM Reference Format:
Fabio Busacca*, Stefano Mangione*, Ilenia Tinnirello*§, Sergio Palazzo§, and Francesco Restuccia†. 2022. SDR-LoRa: Dissecting and Implementing LoRa on Software-Defined Radios to Advance Experimental IoT Research. In *16th ACM Workshop on Wireless Network Testbeds, Experimental evaluation & CHaracterizatiion (WiNTECH '22), October 17, 2022, Sydney, NSW, Australia.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3556564.3558239

1 INTRODUCTION

It is expected that the number of Internet of Things (IoT) devices will surpass 24.1B by 2030, generating up to $1.5T in annual revenue. As the number of IoT devices grows rapidly, both industry and academia are devoting considerable efforts in the design, development, and performance evaluation of IoT-based technologies, with a particular focus on low-power wide-area network (LPWAN) technologies [1–3]. The key differentiator of LPWAN communication protocols is that they trade off high transmission data rates with low-power and long-range communications, thus becoming more suitable than other wireless protocols (e.g., Wi-Fi/LTE) for energy-constrained IoT scenarios, e.g. city-wide pollution and air quality monitoring, among other applications.

Among the numerous available LPWAN protocols, LoRa is among the most promising ones, and has inspired several works in the existing literature [4, 5]. The key challenge is that LoRa is a proprietary protocol, and thus many implementation details are hidden and/or unclear to researchers. Thus, researchers could benefit from a full-fledged reverse-engineering of the protocol, as it would allow to explore its full potential, and even improve its efficiency and performance. Existing Software Defined Radio (SDR)-tailored LoRa implementations [6–10] do not provide a full-fledged transceiver code, either because of lack of reconfiguration capabilities [7], lack of critical functionalities such as frequency shift tracking [8], or lack of transmitter implementation [9].

For this reason, we perform a under-the-hood study of the LoRa physical layer (PHY) and provide a fully reconfigurable SDR implementation of the LoRa PHY, at both transmitter's and receiver's side. As mentioned earlier, SDRs can fully replace real hardware transceivers with software-based functions and therefore pave the way to several novel research contributions, ranging from the study of the limitations and potentialities of the protocol, to the extension, modification, and improvement of the LoRa physical layer. Indeed, an SDR implementation allows access to waveform-level information (i.e., I/Q samples) that cannot be obtained through commercial-level gateways, thus enabling advanced research on topics such as radio fingerprinting, interference cancellation, and adaptive parameter optimization.

We summarize the main contributions of our work:

- We have entirely reverse-engineered the LoRa PHY layer functionalities, including the procedures of packet modulation, demodulation, and preamble detection;
- We have developed the first fully functional software implementation of the LoRa PHY layer for SDR from scratch;
- We have added an extra reliability layer by integrating LoRa with ARQ error-control protocols and rate-less packet-level coding;
- We have evaluated our implementation in two different settings: i) the Colosseum wireless channel emulator, and ii) a real testbed with off-the-shelf USRP radios and commercial devices. The results demonstrate both effectiveness of our implementation, and its interoperability with LoRa commercial devices.

2 RELATED WORK

In this section, we summarize existing work on LoRa SDR implementations. From a theoretical point of view, Bernier et al. [6] offered a complete study of the preamble and start-of-frame synchronization procedure of LoRa, and also focused on the implementation of low-complexity frame synchronization algorithms. The authors provided some performance insights of such algorithms in terms of phase estimation error and synchronization failure probability. However, the implementation of the other components of the LoRa transceiver chain was neglected. Knight and Seeber attempt to implement a full LoRa PHY stack for SDRs in [7]. However, this implementation lacks some functionalities, e.g. the possibility to tune the Spreading Factor (SF) – the only allowed value is 8 – and the Coding Rate (CR), and does not properly implement the whitening functionalities.

Marquet et al. [8] provided a thorough description of the LoRa modulation and demodulation architecture. The implementation is exploited to offer some performance insight on the LoRa technology, such as an evaluation of the Bit Error Rate (BER) as a function of SF and CR. However, the implementation in [8] does not include time and frequency shift tracking for chirp spread spectrum (CSS) modulation, and is therefore unable to decode LoRa signals. Robyns et al. [9] provided an implementation called `gr-lora`, where the authors have reverse-engineered the functionalities of a LoRa receiver. This implementation is therefore successful at decoding LoRa signals generated by commercial devices. However, the transmitter has not been included in the implementation.

Tapparel et al. [10] provide an implementation that includes a Carrier Frequency Offset (CFO) estimation functionality, and is therefore able to communicate with LoRa commercial devices. Moreover, the authors validate their implementation through experiments on USRP SDR hardware. However, such experiments are run on dedicated cables connecting the transmitter to the receiver. Hence, the provided BER values do not include any possible performance degradation resulting from external interference. Finally, the performance analysis results from a fixed configuration, with SF = 7, a bandwidth of 250 kHz, and a payload of 64 bytes.

3 BACKGROUND NOTIONS

LoRa (short for *Long Range*) is a LPWAN proprietary protocol owned by Semtech. LoRa is based on the CSS modulation technology, and supports reliable low data-rate transmissions over long distances, ranging from 1-2 to 10 (and possibly more) kilometers. The actual transmission range and data-rate strongly depend on the SF setting. The SF is an important spectral parameter, and roughly corresponds to the number of chips per bit transmitted. Higher SFs yield a longer transmission range, at the expense of a lower bit-rate, and vice-versa. LoRa specifies the PHY only. As such, it lacks link-layer and networking functionalities, which are instead defined by the LoRaWAN protocol from Semtech. Typical LoRa networks are arranged in a star-of-stars topology, where few LoRa *gateways* collect data transmitted by the LoRa *nodes*. The received data can eventually be forwarded to the Internet thanks to the networking functionalities implemented by LoRaWAN.

LoRa Regional Parameters. LoRa operates in the sub-GHz bands of the Industrial, Scientific, and Medical (ISM) spectrum, according to specific regional frequency plans: the **EU433** and the **EU863-870** bands for Europe, the **US902-928** band for US, and the **AS923** band for Asia. Another region-specific parameter is the supported bandwidth: European countries usually support a single bandwidth of 125 kHz, while US allow the usage of both 125 and 500 kHz. The maximum supported data-rate is influenced accordingly, as a bigger bandwidth guarantees higher transmission rates.

LoRa Transmission Parameters. LoRa supports six different SFs ranging from 7 to 12. However, a SF equal to six is also allowed in some implementations. LoRa also supports up to three different bandwidth configurations of 125, 250, and 500 kHz, respectively. Both parameters can be set to reach the desired trade-off between data rate and reliability. Indeed, higher SFs and smaller bandwidths increase the sensitivity and robustness of the receiver, while lower SFs and bigger bandwidths maximize the transmission data-rate.

Finally, the robustness of LoRa communications is further boosted by the usage of Forward Error Correction (FEC) techniques. LoRa supports four different CR values, according to the formula $4/(4+n), n \in \{1,2,3,4\}$, where n is the number of redundant information bits. A bigger n increases the data protection, but negatively impacts the effective transmission rate.

3.1 Chirp Spread Spectrum (CSS) Modulation

LoRa implements a CSS modulation, which has been demonstrated to be very robust against in-band or out-band interference, which can be very critical when operating in ISM bands. In particular, LoRa employs an M-ary modulation scheme based on chirps [11]. Basic chirps are constant envelope signals whose frequency is linearly modulated sweeping from f_{min} to f_{max} (up-chirp), or from f_{max} to f_{min} (down-chirp). Chirps are cyclically-shifted to produce different symbols, and this cyclical shift carries the information. A symbol, whose length is divided in K equal time intervals called chips, can be cyclically shifted from 0 to $K-1$ positions. The reference position is given by the un-shifted (base) symbol at the beginning of the LoRa frame, which is also used for building the frame preamble.

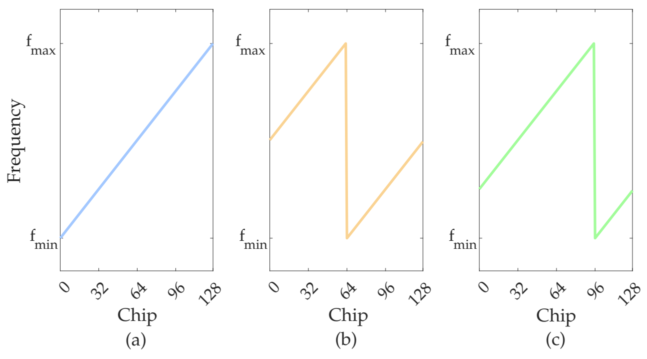

Figure 1: Instantaneous frequency of three upchirp signals for $SF = 7$. The basic upchirp can be shifted to represent up to 2^{SF} symbols, each encoding SF bits. The blue line (a) is the basic upchirp and encodes symbol $M = 0$; the orange line (b) encodes the symbol $M = 64$, while the green line (c) encodes the symbol $M = 96$.

For a given bandwidth $B = f_{max} - f_{min}$, the symbol time depends on the SF parameter, which defines two modulation features: (i) the time duration of each chirp (or, equivalently, the slope of the linear frequency sweep), which is given by 2^{SF} chip intervals; and (ii) the number of raw bits encoded by that symbol, equal to SF. The Data Rate (DR) thus depends on the bandwidth B in Hz, the SF and the Coding Rate (CR) as:

$$DR = SF \cdot \frac{B}{2^{SF}} \cdot CR \tag{1}$$

where $1/B$ is the chip interval, the factor $B/2^{SF}$ provides the symbol rate and the coding rate $CR = 4/(4 + RDD)$ depends on the number of redundancy bits (RDD, from 1 to 4) used for Hamming code forward error correction. The bandwidth can be configured as 125kHz, 250kHz and 500kHz (typically 125kHz is used in the 868MHz ISM band).

Fig. 1 shows the modulating signal used for (i.e. the instantaneous frequency of) a basic upchirp and two examples of circular shifts obtained for $SF = 7$: the symbol time is $T = 128T_c$, while the two exemplary shifts encode the symbols 64 and 96, respectively. The instantaneous frequency of an unmodulated (base) LoRa chirp can be written as:

$$f_i^{(0)}(t) = -\mu\frac{B}{2} + \mu\frac{B}{T}t \tag{2}$$

where $\mu = +1$ gives an *up*chirp and $\mu = -1$ a *down*chirp, $T = 2^{SF}T_c$ is the symbol time and $T_c = 1/B$ the chip duration, $0 \le t < T$. There are $K = 2^{SF}$ possible symbols, each representing a cyclic shifted version of the base upchirp. The instantaneous frequency of symbol k is thus given by:

$$f_i^{(k)}(t) = \begin{cases} +\mu\frac{B}{2} + \mu\frac{B}{T}(t - kT_c) & 0 \le t < kT_c \\ -\mu\frac{B}{2} + \mu\frac{B}{T}(t - kT_c) & kT_c \le t < T. \end{cases} \tag{3}$$

The LoRa preamble starts with several repetitions of a base upchirp:

$$f_{pr}(t) = -\frac{B}{2} + B\left(\frac{t}{T} - \left\lfloor\frac{t}{T}\right\rfloor\right). \tag{4}$$

After several consecutive base upchirps, the preamble features two modulated symbols, called *sync* words, for network identification, and 2.25 downchirps which are useful for accurate synchronization. Overall, the preamble of a LoRa frame is constituted by a sequence of at least eight upchirps (including the two modulation sync words), followed by two and a quarter downchirps. Following the preamble, the payload header, the payload and an optional frame check sequence are transmitted by using the cyclically-shifted M-ary modulation.

3.2 Receiver Sensitivity

Another fundamental parameter in LoRa modulation is the receiver sensitivity, i.e. the minimum detectable signal strength. According to [12], the sensitivity for a LoRa receiver is:

$$S = -174 + 10\log_{10} BW + NF + SNR \ [dBm] \tag{5}$$

Where the first term is the thermal noise power in 1 Hz of bandwidth at a room temperature of 300K; BW is the transmission bandwidth; NF is the receiver noise figure, and is typically equal to 6 dB for popular transceivers models, such as Semtech SX1272 or Semtech SX1276 [11]. The last term is the Signal-to-Noise Ratio (SNR) value required at the receiver input for a successful demodulation, and depends, once again, on the receiver architecture, and on the SF, too. Typical values for SNR are reported in Table 1.

SF	7	8	9	10	11	12
$SNR[dB]$	-7.5	-10	-12.5	-15	-17.5	-20

Table 1: SNR thresholds for several SF values.

According to (5), the sensitivity is influenced by both the bandwidth and the SF settings. Hence, high SF values and small bandwidths achieve a high receiver sensitivity, at the expense of low data-rates. Conversely, faster LoRa communications are associated to low SFs, and larger bandwidths, but usually suffer of a low sensitivity. In other words, SF and BW can be properly tuned to achieve the desired trade-off between reliability and data rate.

4 SDR LoRa TRANSCEIVER

In this section, we present the design and implementation of our LoRa transceiver for SDR platforms, together with some preliminary simulation results on the transceiver performance. All simulations have been run in Matlab. The receiver is based on three main modules: (i) the symbol detection module, which is responsible of identifying the symbols encoded in the received signal, once a new packet is correctly identified and synchronized; (ii) the synchronization module, which identifies the beginning of a new packet, and estimates the carrier and timing references used by the transmitter; (iii) the (optional) drift tracking module that compensates the clock drifts between transmitter and receiver for the correct demodulation of long frames. Apart from the building of LoRa symbols, we also implemented a pipeline of processing operations in the TX chain, including: parity check coding, whitening, shuffling and interleaving, and Gray coding.

4.1 Symbol Detection

LoRa demodulation can be implemented with very simple operations by mapping in each symbol the time interval at which the chirp jumps from f_{max} to f_{min} in a easily detectable frequency. In particular, our implementation works as follows. First, each received symbol is multiplied with the synchronized base down-chirp (at the same SF of the received signal). The result is a signal comprising only two frequencies (namely, $-k/T$ and $-B - k/T$, with T being the symbol time) which depend on the transmitted symbol k. Second, by down-sampling the signal at the rate B, both frequencies can be aliased to the same frequency $-k/T$. Finally, the signal is transformed in the frequency domain by means of an FFT. The symbol index k can be estimated by considering the position of the peak at the output of the FFT.

An interesting feature of LoRa is the quasi-orthogonality of signals transmitted at different SFs, as well as the robustness against external interference sources. Indeed, after the multiplication of the signal with the synchronized downchirp, the interfering signal is mapped into a noise over the whole frequency band of the signal, which prevents the correct identification of the symbol peak only for very low SIR (Signal-to-Interference Ratio) values. Conversely, when the interfering signal is a LoRa signal at the same SF, the receiver will observe multiple peaks at the output of the FFT: a maximum peak corresponding to the reference symbol, and two smaller peaks corresponding to two - partially overlapping - interference symbols. In such a case, if the reference signal is a few dB stronger than the interfering one, the symbol can be detected.

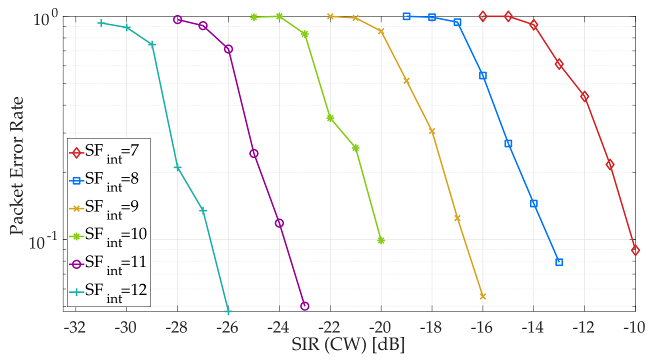

Figure 2: PER vs *SIR* in case of sinusoidal interference.

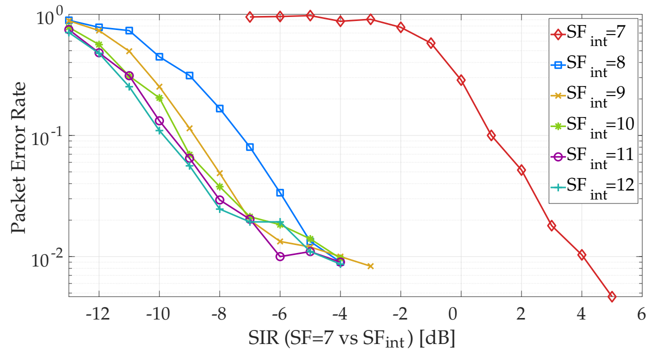

Figure 3: PER vs *SNR* values in case of interference with other LoRa modulated signals.

Fig. 2 shows the performance of our receiver in terms of Packet Error Rate (PER) in case of interference with an in-band sinusoidal (or narrow-band) signal. The reference signal is given by the same packet with a payload of 20 bytes, which has been modulated at different SFs as indicated in the labels. From the figure, we observe that there are about 3dB of difference between the *SIR* values which guarantee a target *PER* for signals modulated at SF x and $x + 1$. Fig. 3 quantifies the robustness of our receiver against interference sources generated by other LoRa signals. The figure depicts the *PER* versus the *SIR* values for a reference LoRa signal modulated at SF 7, with $B = 125$ KHz and a payload of 20 bytes, in case of interference with LoRa signals modulated under different SFs (as indicated in the labels). As expected, the interference due to a LoRa signal modulated at SF 7 is critical, because even a *SIR* value of 1 dB (often called *capture threshold*) can result in a PER equal to 0.1. For the other signals modulated at different SFs, it is evident that the signal orthogonality is not perfect: when the interfering signal is about 10dB stronger than the reference one, the PER is higher than 0.1. This SF-dependent threshold is often called *interference rejection* threshold. Obviously, the receiver has to correctly identify the SF used by the transmitter and the boundary of each symbol, as detailed in the next section.

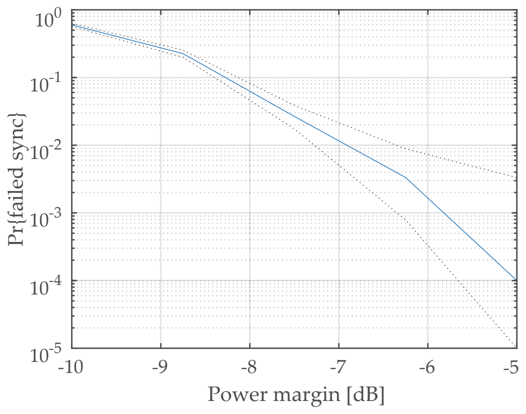

Figure 4: Probability of failing to detect a preamble.

4.2 Carrier and Time Synchronization

In order to recognize the symbol boundary, the receiver has to first identify the exact beginning of a preamble in time and in frequency, as well as the symbol duration corresponding to the correct SF.

Our synchronization mechanism is built by exploiting the preamble structure, which includes both upchirp and downchirp transmissions. The idea is mixing (i.e., multiplying) the received signal $f_{rx}(t)$ with the complex conjugate of a reference preamble upchirp $f_{pr}(t)$. Since the downchirp has the same absolute slope as the unsynchronized upchirp of the preamble, the frequency of the mixed signal $f_{mix,1}(t) = f_{rx}(t) - f_{pr}(t)$ changes over time as $\lfloor (t-\tau)/T \rfloor - \lfloor t/T \rfloor$, which is a square wave with values 0 and ± 1 and duty cycle $|\tau|/T$. It follows that the output of the mixer signal features tones at only two frequencies $\nu_1 = \text{CFO} - B\tau/T$ (when $t \geq \tau$) and $\nu_1 \pm B$ (when $t < \tau$). The same mixing mechanism can be applied to the last part of the preamble (constituted by downchirps), by multiplying the signal with base upchirps. The resulting signal has the same structure as the previous one with frequencies $\nu_2 = \text{CFO} + B\tau/T$ and $\nu_2 \pm B$. If ν_1 and ν_2 are available, the estimated carrier offset CFO_{est} can be computed as the average between ν_1 and ν_2, while the estimated timing offsets τ_{est} can be computed as $T/B \cdot (\nu_2 - \nu_1)/2$.

In order to identify a new preamble, the receiver:

(1) samples the received signal $r(t)$ with a sampling frequency $f_s = B \cdot \text{OSF}$, i.e., Over Sampling Factor (OSF) times the nominal bandwidth of the signal, obtaining $r_n = r(n/f_s)$
(2) multiplies (mixes) a window of $N = K \cdot \text{OSF}$ samples with a base downchirp, obtaining:

$$z_n = r_n \exp(\jmath\pi(n/\text{OSF} - n^2/(N \cdot \text{OSF}))) \tag{6}$$

(3) computes the absolute value of the FFT of signal z_n:

$$\Gamma_k = \left| \sum_{n=0}^{N-1} z_n \exp(-\jmath 2\pi nk/N) \right| \tag{7}$$

(4) estimates $\hat{k}$ as the position of the maximum in Γ_k;

Finally, if the estimated position $\hat{k}$ is detected continuously for a number of times (e.g., 3 consecutive windows), the receiver understands that an incoming preamble is received.

This procedure is executed continuously, for each possible SF, even when the demodulation of a frame is already in progress. This

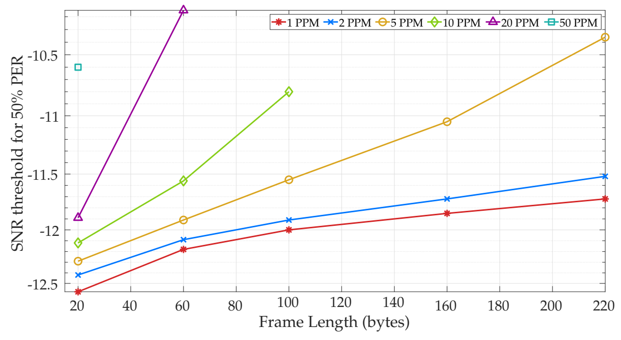

Figure 5: *SNR* values guaranteeing *PER* ≤ 0.5.

algorithm succeeds in detecting a preamble even several dBs *below* the sensitivity threshold. The probability of failing the detection of a preamble is shown in Fig. 4, when the power margin is computed as the difference between the received signal power and the receiver sensitivity. From the figure, it is evident that the receiver is able to detect a preamble even when the received power is below the sensitivity threshold of -121 dBm. Only when the margin is smaller than 7 dB, is the failure probability higher than 1%. Once the preamble is detected, fine estimates of v_1 and v_2 can be obtained as follows:

$$\hat{v} = \frac{B}{K}\left(\hat{k} + \frac{1}{2}\frac{\Gamma_{\hat{k}-1} - \Gamma_{\hat{k}+1}}{\Gamma_{\hat{k}-1} - 2\Gamma_{\hat{k}} + \Gamma_{\hat{k}+1}}\right) \tag{8}$$

where the values of $\hat{k}$ and Γ_k are obtained from the multiplication with the base downchirp in the first part of the preamble, and with its conjugate for the last 2.25 preamble symbols (for the last portion of the preamble, made of downchirps). Equation (8) is based on a parabolic interpolation around the maximum, and yields excellent results even for very low SNR values. For a reference signal transmitted at SF 7 and power margin of -10dB, the standard deviation of the CFO estimation error σ_{Cfo} over a bandwidth of 125 kHz is about $9 \cdot 10^{-4}$, while for a power margin of -5dB the ratio σ_{Cfo}/BW is reduced to $6.8 \cdot 10^{-4} \cdot BW$.

4.3 Impact of Clock Drifts

Once a preamble has been detected and initial estimates of CFO and τ have been performed, the LoRa receiver should periodically update these estimates for compensating the clock drifts between the transmitter and the receiver. Most SDR implementations have not addressed this issue, because SDR platforms usually rely on clocks whose errors are in the order of 1 PPM. However, off-the-shelf devices may experience inaccuracy can be as high as 17 PPM, due to low-cost crystal oscillator which may lead to synchronization problems (especially for long frames).

We quantified the SNR threshold leading to a PER lower than 0.5 for a receiver not implementing any clock tracking mechanism and for different packet transmission times. Specifically, Fig. 5 shows the SNR values as a function of the frame length, for a reference signal modulated at SF 7 and $B = 125$ kHz, for different clock stability values. Irrespective that the reference signal is transmitted at SF 7 (and therefore the transmission times are accordingly minimized), packet reception is completely prevented for a clock frequency error higher than 10 PPM and a frame length higher than 60 bytes (PPM 50) or 100 bytes (PPM 20). Otherwise, packet reception works, but the errors are relevant.

A possible solution is extending the receiver architecture with the clock tracking module, such as the one we designed in [13]. Once a window of consecutive symbols is demodulated, the signal regenerated at the receiver is correlated with three different versions of the originally received signal: the one obtained considering a time offset equal to the initial estimate τ_{est}, and two other versions in which the offsets are equal to $\tau_{est} \pm 1/f_s$ (being f_s the sampling frequency), i.e., shifted by plus or minus one sample. The three correlation operations will result in three different maximum values, which will be interpolated using a quadratic function. Finally, the maximum of this parabolic interpolation will provide the time offset used in the current window for compensating the clock drift. In our implementation, we chose a time window of four symbols as a trade-off of accuracy and complexity.

4.4 Transmitter Implementation

At the transmitter side, we implemented parity check coding, whitening, shuffling, interleaving, and gray coding, to increase robustness towards synchronization errors or narrowband interference, which can be a serious issue for CSS-based modulations. While most of the transmitter side operations have been implemented as described in the LoRa modulation patent [12], a few implementation details required a low-level analysis of real signals, transmitted by LoRa commercial devices. Indeed, the patent leaves some ambiguities (for example about the row to column ordering in the interleaving, or the actual parity check matrix for coding at rate 4/5 and 4/8). Moreover, the initialization values for the CRC computation in the payload is not constant as specified in the patent, but rather seems to depend on the payload size. We found the initialization vectors for each possible packet size by exhaustive search. The resulting implementation has been demonstrated to be compatible with commercial devices.

5 IMPROVING LINK RELIABILITY

Taking into account that each device experiences a specific Packet Delivery Rate (PDR) as a function of its channel and interference conditions, we designed and implemented two different solutions for improving link reliability: i) the usage of on-demand retransmissions of corrupted frames, by means of an Automatic Repeat Request (ARQ) protocol, ii) the proactive transmission of additional frames, generated by means of rate-less coding schemes applied at a frame-level. Obviously, the first approach is effective in case a generic device is involved in sporadic transmissions, while the second one is useful when devices need to transmit a burst of multiple frames.

ARQ Mechanism. We implemented an optional Automatic Repeat Request (ARQ) reliability mechanism, to integrate and complement the FEC Hamming techniques natively implemented by the protocol. To support packet re-ordering and acknowledgment, as well as addressing, we reserve the first three bytes of the LoRa packet payload for the **Destination**, **Source**, and **Sequence Number** fields. Our protocol has several steps: (i) Transmission of the actual data, (ii) Response from the receiver with a NACK message,

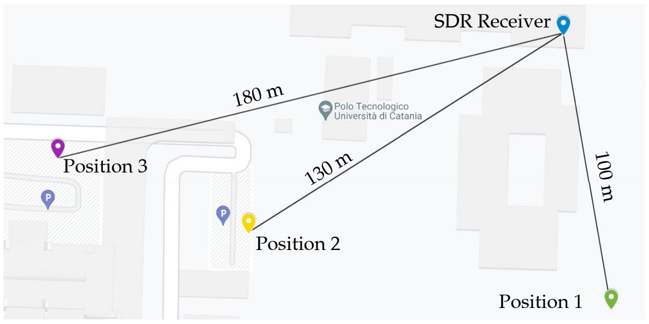

Figure 6: Map of the testbed locations.

(iii) Re-transmission of the missing packets, if needed, (iv) Transmission of a final ACK from the receiver, and (v) Termination of the communication phase. To avoid any deadlock, during (ii), if no response is received from the transmitter for a specified timeout, the receiver re-sends the NACK packet.

Rateless Coding. Rateless codes such as those described in RFC6330 work by exploiting the possibility of generating as many encoded frames as needed from a burst of k frames at the transmitted side. Indeed, thanks to the coding scheme, the receiver is able to decode an exact copy of the entire burst of k frames from any subset of $k + v$ successful received (i.e. non-erased) encoded frames. The required reception overhead $\epsilon = v/k$ is usually in the order of a few percentage points. An interesting feature of this solution is that no downlink channel is required as a feedback to transmitter. Being downlink bandwidth limited in LoRaWAN networks, such an approach can be useful in many practical scenarios. We implemented a signalling mechanism from the network server to devices, for notifying the device-specific PDR. On the basis of this value and a target probability γ of successful delivery for a burst of k frames, each device involved in the transmission of data bursts computes the number k' of coded frames to be generated from each group of k frames, as:

$$k' : \sum_{l=k+v}^{k'} \binom{k'}{l} PDR^{l} \cdot (1 - PDR)^{k'-l} \geq \gamma \tag{9}$$

The expected number of delivered frames is $k' \cdot PDR$. The usage of rateless coding increases the load offered to the network and, consequently, the PDR experienced by other devices. However, when the number of devices involved in the transmission of data bursts is limited or when the main source of frame losses is the channel, the load increment has a minimal effect on the PDR variations and a single iteration suffices for finding a stable k'.

6 EXPERIMENTAL RESULTS

We validated the implementation of our LoRa transceiver by running some compatibility tests with commercial LoRa devices. We set one SDR platform running the LoRa transceiver in our laboratory at the University of Catania, and deployed the devices in outdoor in locations characterized by partially obstructed links as depicted in Fig. 6. In position 1, where nodes were perfectly visible, we got a Packet Delivery Rate (PDR) equal to 1 for each available SF; in position 2, some errors were found at SF 7, while in position 3, the PDR was lower than 0.1 at SF 7 and about 0.5 at SF 10.

In order to study the performance of our prototype, we also integrated the LoRa transceiver on the Colosseum testbed, where multiple coordinated SDR platforms and channel emulators are available. An important feature of the testbed is the possibility of controlling the network scenarios, in terms of channel models between nodes, as well as the transmission patterns of coexisting devices. Indeed, the transmission attempts of independent SDR platforms can be synchronized or shifted of a desired time interval, for characterizing specific interference effects generated by multiple interference sources. Moreover, the testbed allows the analysis of low-level signals collected by independent receivers, enabling many experimental studies on receiver architectures.

6.1 Testbed Description

We evaluated SDR-LoRa on Colosseum, the world's largest network emulator [14]. Colosseum is a wireless emulator with 128 so-called Standard Radio Nodes (SRNs), i.e. dedicated hardware nodes each equipped with a NI/Ettus USRP X310 SDR. Each SRN can host and run user-defined Linux Containers (LXCs), to offer a high degree of freedom in the customization and usage of the underlying hardware. The SRNs are all linked together by the Colosseum Massive Channel Emulator (MCHEM). The latter is made up of several FPGA modules and is thus able to process the radio signals through Finite Impulse Response (FIR) filters. The MCHEM can therefore emulate the effects of real-world wireless RF channels, such as attenuation, propagation delay, fading, and multipath. Another fundamental module of the Colosseum architecture is the RF scenario server. A Colosseum scenario is a collection of wireless links between several SRNs, where each link is defined by digital channel taps. When a scenario is activated on the emulation platform, these channel taps are fed to the MCHEM at run time. Our results are based on a custom LXC image. This container includes the developed SDR implementation, together with the libraries and system tools needed to run our code. **We pledge to share the full LXC container with the community, to allow reproducibility and experimentation with our code.**

6.2 Results for Single Link

We run a first set of experiments with a single LoRa link, with two SDR nodes acting as a transmitter and receiver, connected by means of the MCHEM channel emulator. The selected RF scenario is characterized by a noise power with $\sigma_n = 3.5 * 10^{-8}$ and a tunable Path Loss.

Channel Attenuation	45 dB	50 dB	55 dB	60 dB
PDR	1	0.99	0.74	0.20

Table 2: PDR vs Channel Loss.

Table 2 shows the PDR at the receiver, as a function of different channel attenuation values[1], when the receiver transmits at SF 7, with a bandwidth of 125kHz, a payload size of 50 bytes, and a fixed normalized amplitude of the signal equal to 1. The link also suffers of additional power losses, due to the connectors between SDR nodes and the wired channel emulator. Since these losses cannot be

[1]This choice is useful to emulate several transmission distances. In fact, in free-space communications, a larger path-loss corresponds to a longer communication distance.

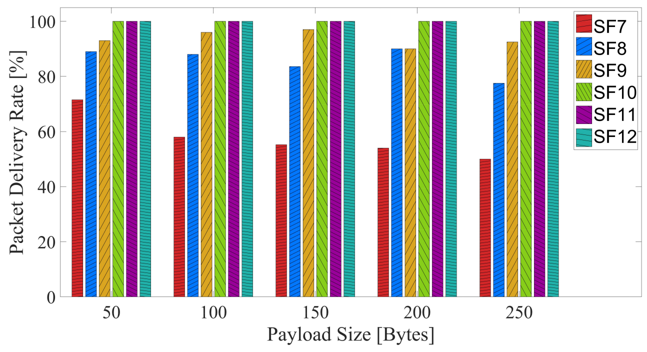

Figure 7: PDR vs the packet Payload Size.

easily quantified, we experimentally found the channel attenuation which results in an *SNR* lower than the reception threshold. Indeed, the *PDR* is lower than 1 (about 75%) for a channel attenuation of 55dB and lower than 15% when the channel attenuation is increased to 60dB.

For a channel attenuation value of 56dB (leading to an SNR value lower than the reception threshold at SF 7), we run further experiments at different SFs and for different payload sizes. We measured the *PER* when transmitting a fixed amount of data (namely, 10000 bytes), which corresponds to a different total number of frames. The results are summarized in Fig. 7. From the figure, it is evident that the SNR value at the receiver is higher than the minimum reception thresholds for SFs 10, 11 and 12. For the other SFs, for which the correct demodulation of the symbols cannot be guaranteed, the *PDR* is affected by the payload size, with a general degradation as the payload size increases (although there are a few exceptions due to the confidence interval of the results). However, the *PDR* does not decrease as an exponential function (with a base lower than 1) of the number of symbols in the frame, which are roughly proportional to the payload size. This suggests that synchronization problems can be the main cause of packet losses for the *SNR* values considered in the experiments.

On top of error-prone links, we also tested our schemes for improving link reliability. Fig. 8 quantifies the total amount of time needed to completely transfer 10000 bytes of data to the receiver, for a channel attenuation of 56 dB. The time takes into account the total interval required for transmitting all the frames, including selective retransmissions of corrupted frames. No duty cycle has been considered. Note how higher SFs exhibit a bigger transmission time despite the fact that the links are more robust: indeed, the higher is the SF, the higher is the number of chips in a single LoRa symbol, and thus, the bigger is the transmission time of each frame.

An alternative approach for improving the link reliability is exploiting rate-less coding. Table 3 summarizes the number of coded frames required for guaranteeing a delivery probability $\gamma \leq 1 - 10^4$ and the relevant transmission time, for the same scenario of a data transfer of 10000 bytes, a channel attenuation of 56 dB, and a payload size equal to 100 or 250 bytes. The table refers to the usage of Raptor-Q codes, assuming to use $\nu = 5$ (which guarantees according to RFC6330 a probability lower than 10^{-10} that the frame decoding will fail). For a given *PDR*, the number of additional packets generated by the usage of rate-less coding is obviously

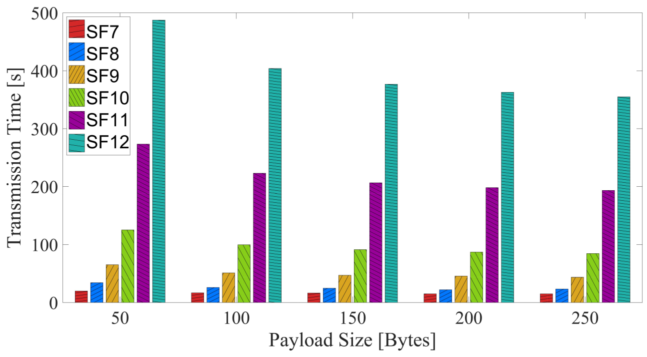

Figure 8: Transmission vs the packet Payload Size.

greater than the one resulting from selective retransmissions, but the difference gets lower as the the total number of packets to be sent increases (for example, for 100 bytes and $SF = 9$ we transmit 125 packets, with a total number of expected transmissions equal to 108 packets).

	100 bytes			250 bytes		
SF	PDR	k'	Time	PDR	k'	Time
7	0.58	220	41.6 s	0.48	129	46.3 s
8	0.84	142	48.0 s	0.78	72	46.5 s
9	0.92	125	76.9 s	0.92	57	65.4 s

Table 3: Overheads of Raptor-Q coding

6.3 Results for Multiple Links

We run a second set of experiments for studying the impact of interference between multiple coexisting LoRa links. First, we analyzed the imperfect orthogonality of different SFs. We set-up a reference LoRa link working at *SF* 7 and transmitting frames with a payload of 50 bytes, and an interfering node working at a *SF* different from *SF* 7. The interfering node has been configured for transmitting continuously, without any duty cycle, in order to guarantee that all the frames transmitted by the reference link overlap in time with the interfering node, thus leading to collisions. Fig. 9 shows the *PDR* measured in our experiments, when the transmitted frames collide with interfering signals at different *SF*s, as a function of the *SIR*. The limits of imperfect orthogonality are quite evident: for *SF* 8, it is enough a *SIR* value of -4 dB for damaging the frames and reducing the *PDR* to about 75%. Although the interference generated at *SF* 8 is the one reducing the *PDR* the most, we also notice that the *SIR* threshold which prevents a *PDR* equal to 1 is about the same for all the *SF*s used by interfering node.

The large rejection thresholds of the interfering signals is due to the receiver operation. When multiplying the received signal with the downchirp at *SF* 7 and computing the *FFT*, such an interfering power is spread on the whole bandwidth of 125kHz, while the power of the reference signal is seen as a narrow peaks whose position within the band corresponds to the coded symbols.

Obviously, a possible cause of symbol detection error due to a low *SIR* value is the presence of multiple interfering signals. Fig. 10 quantifies the *PDR* achieved by the reference link at *SF* 7, when multiple interfering nodes are active at *SF* 8 and for different normalized amplitude values of the reference signals. We let this

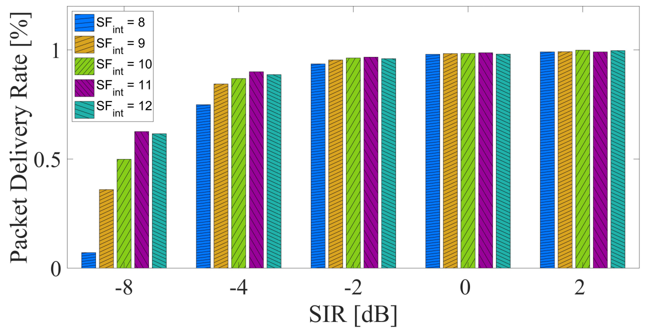

Figure 9: PDR vs *SIR* for a reference link at *SF* 7, in presence of collisions with different *SF*s.

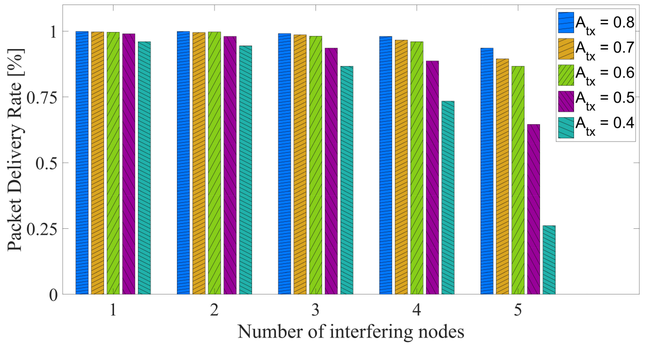

Figure 10: PDR of a reference link at *SF* 7 as a function of the number of interfering nodes at *SF* 8.

amplitude value vary in the range 0.4-0.8, while the normalized amplitude of the interfering signals is fixed and equal to 0.5.

We also considered the effects of collisions generated at *SF* 7. In principle, signals transmitted at the same *SF* are not orthogonal and should prevent the correct reception of the colliding frames. However, the capture thresholds of LoRa are very low: it is enough that the reference link has a *SIR* of a few dB (typically, 3 dB) to allow the correct demodulation of the frame. We repeated the experiment with multiple interfering nodes by configuring the interfering signal at *SF* 7. Fig. 11 shows the *PDR* observed as the number of interfering nodes increases and when the transmission power of the reference link is $6dB$ stronger than the interfering ones. From the figure we observe an interesting phenomenon: the *PDR* is almost one even in presence of 4 nodes, when the *SIR* is 0 dB, and slightly lower than 1 in case of 5 interfering nodes (for a negative *SIR*). This is due to the receiver operation, according to which the interference generated by multiple transmitters at *SF* 7 is not additive.

7 CONCLUSIONS

In this paper we described a full-fledged implementation of a LoRa transceiver for SDR platforms. The prototype has been fully integrated into the Colosseum testbed and is ready to be released to the community for enabling access to low-level LoRa signals, thus fostering advanced research on the topic.

We run several experiments for characterizing the receiver performance, in terms of robustness to synchronization and clock

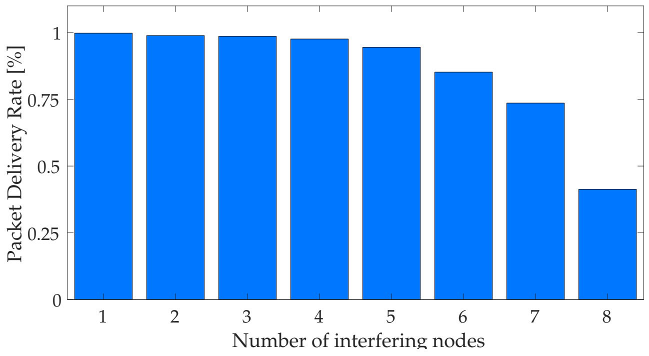

Figure 11: PDR of a reference link at *SF* 7 as a function of the number of interfering nodes on the same *SF*.

drifts errors, as well as to endogenous and exogenous interference sources. Differently from previous studies, where interference generated by multiple nodes was usually emulated by means of a single SDR platform, our testbed allows the configuration and control of multiple and independent interfering sources. Experimental results demonstrated that clock drifts may have a relevant impact when LoRa is used by simple devices, with cheap oscillators, and the benefits of using a clock tracker. We also analyzed the impact of complementary approaches for improving link-level performance, based on selective retransmissions or frame-level coding. While the ARQ scheme is more effective with sporadic traffic, rate-less coding is more suitable for burst transmissions and for high-density networks.

REFERENCES

[1] J. P. S. Sundaram *et al.*, "A Survey on LoRa Networking: Research Problems, Current Solutions, and Open Issues," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 371–388, 2019.
[2] A. Ikpehai *et al.*, "Low-power Wide Area Network Technologies for Internet-of-Things: A Comparative Review," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2225–2240, 2018.
[3] U. Raza *et al.*, "Low Power Wide Area Networks: An Overview," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 855–873, 2017.
[4] A. Al-Shawabka *et al.*, "DeepLoRa: Fingerprinting LoRa Devices at Scale Through Deep Learning and Data Augmentation," in *Proceedings of ACM Mobihoc 2021*, pp. 251–260, 2021.
[5] D. Garlisi *et al.*, "Interference Cancellation for LoRa Gateways and Impact on Network Capacity," *IEEE Access*, vol. 9, pp. 128133–128146, 2021.
[6] C. Bernier *et al.*, "Low Complexity LoRa Frame Synchronization for Ultra-Low Power Software-Defined Radios," *IEEE Transactions on Communications*, vol. 68, no. 5, pp. 3140–3152, 2020.
[7] M. Knight *et al.*, "Decoding LoRa: Realizing a modern LPWAN with SDR," in *Proceedings of the GNU Radio Conference*, vol. 1, 2016.
[8] A. Marquet *et al.*, "Towards an SDR implementation of LoRa: Reverse-engineering, demodulation strategies and assessment over Rayleigh channel," *Computer Communications*, vol. 153, pp. 595–605, 2020.
[9] P. Robyns *et al.*, "gr-lora: An efficient LoRa decoder for GNU Radio." https://github.com/rpp0/gr-lora, 2017.
[10] J. Tapparel *et al.*, "An Open-Source LoRa Physical Layer Prototype on GNU Radio," in *2020 IEEE SPAWC*, pp. 1–5, 2020.
[11] Semtech Corporation, "LoRa™ Modulation Basics," may 2015.
[12] Semtech, "SX1272/3/6/7/8: LoRa Modem - Designer's Guide," may 2013.
[13] D. Garlisi *et al.*, "Interference cancellation for lora gateways and impact on network capacity," *IEEE Access*, vol. 9, pp. 128133–128146, 2021.
[14] L. Bonati *et al.*, "Colosseum: Large-scale wireless experimentation through hardware-in-the-loop network emulation," in *2021 IEEE DySPAN*, pp. 105–113, IEEE, 2021.