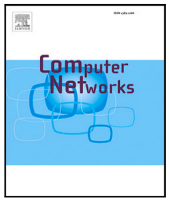
Computer Networks 241 (2024) 110194

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



SDR−LoRa, an open-source, full-fledged implementation of LoRa on Software-Defined-Radios: Design and potential exploitation ☆

Fabio Busacca [b,*], Stefano Mangione [a], Sergio Palazzo [b], Francesco Restuccia [c], Ilenia Tinnirello [a,b]

[a] Department of Engineering, University of Palermo, Italy
[b] Department of Electrical, Electronics and Computer Engineering (DIEEI), University of Catania, Italy
[c] Institute for the Wireless Internet of Things, Northeastern University, Boston, USA

ARTICLE INFO

Keywords:
Software Defined Radio
LoRa
Long range
Flexibility
IoT
LPWAN
Localization
MAC
PHY
Software
Open source
GitHub
Interference cancellation
Protocols
Physical layer

ABSTRACT

In this paper, we present SDR−LoRa, an open-source, full-fledged Software Defined Radio (SDR) implementation of a LoRa transceiver. First, we conduct a thorough analysis of the LoRa physical layer (PHY) functionalities, encompassing processes such as packet modulation, demodulation, and preamble detection. Then, we leverage on this analysis to create a pioneering SDR-based LoRa PHY implementation. Accordingly, we thoroughly describe all the implementation details. Moreover, we illustrate how SDR−LoRa can help boost research on the LoRa protocol by presenting three exemplary key applications that can be built on top of our implementation, namely fine-grained localization, interference cancellation, and enhanced link reliability. To validate SDR−LoRa and its applications, we test it on two different platforms: (i) a physical setup involving USRP radios and off-the-shelf commercial devices, and (ii) the Colosseum wireless channel emulator. Our experimental findings reveal that (i) SDR−LoRa performs comparably to conventional commercial LoRa systems, and (ii) all the aforementioned applications can be successfully implemented on top of SDR−LoRa with remarkable results. The complete details of the SDR−LoRa implementation code have been publicly shared online, together with a plug-and-play Colosseum container.

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.

LoRa stands out as one of the most promising LPWAN protocols, 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 [6,7]. Therefore, researchers could benefit from an open-source, full-fledged reverse-engineering of the protocol, as it would allow to explore its full potential, and even improve its efficiency and performance.

Many existing SDR-tailored LoRa implementations [8–13] do not provide a full-fledged transceiver code, because of either lack of reconfiguration capabilities [9], lack of critical functionalities such as frequency shift tracking [10], or lack of transmitter implementation [11].

To this purpose, we here conduct an in-depth examination of the underlying mechanisms of the LoRa PHY and present a completely reconfigurable SDR implementation of the LoRa PHY for both transmission and reception aspects. As previously mentioned, SDRs have the capability to entirely supplant physical hardware transceivers with software-driven functionalities, so opening up pathways for various

☆ The work of Fabio Busacca, Stefano Mangione, Sergio Palazzo and Ilenia Tinnirello was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE0000001 - program "RESTART"). The work of Francesco Restuccia was partially supported by the National Science Foundation (NSF) grant CNS-2134973, CNS-2120447 and ECCS-2229472, by the Air Force Office of Scientific Research under contract number FA9550-23-1-0261, and by the Office of Naval Research under award number N00014-23-1-2221.

* Corresponding author.
E-mail address: fabio.busacca@unict.it (F. Busacca).

https://doi.org/10.1016/j.comnet.2024.110194
Received 15 August 2023; Received in revised form 10 January 2024; Accepted 15 January 2024
Available online 19 January 2024

innovative 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.

Actually, a few open-source implementations of LoRa on SDR have been recently developed and presented in the literature. As outlined in Section 2 about Related Works, our implementation differs from other existing ones in several featuring aspects.

We hereby 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;
- Starting from our previous work in [14], we have further improved the first fully functional software implementation of the LoRa PHY layer for SDR, and made it **publicly available on GitHub**[1];
- We have leveraged on our implementation to access low-level information from the LoRa synchronization block and exploited that information for fine-grained node localization;
- We have extended the receiver architecture to support multi-user detection based on successive interference cancellation;
- 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) a real testbed with off-the-shelf USRP radios and commercial devices, and (ii) the Colosseum wireless channel emulator. In the first case we successfully tested the interoperability of our implementation with LoRa commercial devices. The second setting, instead, demonstrated the effectiveness of our implementation and of its applications.

The rest of the paper is organized as follows. Section 2 reviews the existing work on LoRa SDR implementation. Section 3 provides some background notions on the LoRa protocol, useful to understand underlying mechanisms and advantages of the protocol. Section 4 thoroughly describes the design and implementation of SDR-LoRa. Section 5 describes three exemplary applications of SDR-LoRa, namely localization, interference cancellation, and improvement of the network reliability. Section 6 presents several experimental results on the basic SDR-LoRa functionalities, as well on the three addressed applications of SDR-LoRa. Finally, Section 7 summarizes the insights obtained from our experiments, draws some conclusions on our work, and offers some useful hints about future developments stemming from SDR-LoRa.

2. Related work

In this section, we summarize existing work on LoRa SDR implementations. From a theoretical point of view, Bernier et al. [8] 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 [9]. However, this implementation lacks some functionalities, e.g. the possibility to tune the SF – the only allowed value is 8 – and the Coding Rate (CR), and does not properly implement the whitening functionalities.

Marquet et al. [10] 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 [10] 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. [11] 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. [12] 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.

Finally, an alternative full-fledged and open-source implementation of LoRa on SDR is presented in [13]. However, their work is based on the Matlab language, and does not feature real-time reception and transmission of LoRa signals.

All in all, our implementation solves all the shortcomings of the above mentioned works, as it offers a full-fledged implementation of the LoRa stack. Moreover, SDR-LoRa is written in the more versatile Python language with the support of the C-based numpy library, and is therefore extremely efficient. As specifically compared to [13], SDR-LoRa also includes a native interface with SDR radio, to support plug-and-play experiments with real life hardware.

3. Background notions

In this section, we provide some important background notions on the LoRa protocol, useful to understand the underlying mechanisms and the main features of the protocol, as well as the applications discussed in Section 5; for instance, the mathematical background of LoRa is useful to understand the issues caused by a timing drift. The latter is of crucial importance to achieve the goal of interference cancellation addressed in Section 5.2.

3.1. LoRa in a nutshell

LoRa (short for *Long Range*) is a LPWAN proprietary protocol owned by Semtech [15]. 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 , proportional to the energy spent for each transmitted bit.

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.

[1] https://github.com/fabio-busacca/sdr-lora

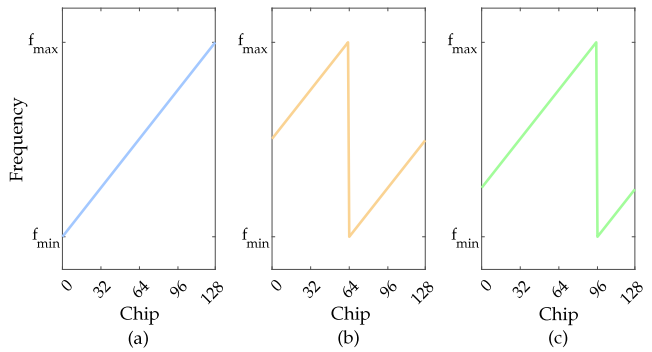


Fig. 1. Instantaneous frequency of three upchirp signals for $SF = 7$. The basic upchirp can be shifted to represent up to 2^{SF} symbols, each encoding SF bits. The blue line (a) is the basic upchirp and encodes symbol $M = 0$; the orange line (b) encodes the symbol $M = 64$, while the green line (c) encodes the symbol $M = 96$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

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.

3.2. Chirp spread spectrum (CSS) modulation

LoRa implements a CSS modulation [16], 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 [17,18]. 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.

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:

$$DR = (SF - 2 \cdot (SF > 10)) \cdot \frac{B}{2^{SF}} \cdot CR \tag{1}$$

where the boolean condition evaluates to 1 if true and 0 if false, $1/B$ 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 125 kHz, 250 kHz and 500 kHz (typically 125 kHz is used in the 868MHz ISM band).

Table 1
SNR thresholds for several SF values.

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

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.

3.3. Receiver sensitivity

Another fundamental parameter in LoRa modulation is the receiver sensitivity, i.e. the minimum detectable signal strength. According to [19], the sensitivity for a LoRa receiver is:

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

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 [16]. The last term is the 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 [20].

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 outline the design and implementation of our LoRa transceiver tailored for SDR platforms, along with preliminary simulation results showcasing the transceiver's performance. All simulations were conducted using MathWorks Matlab.

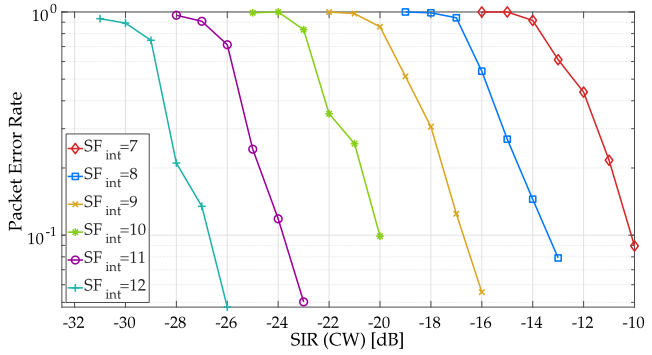


Fig. 2. PER vs SIR in case of sinusoidal interference.

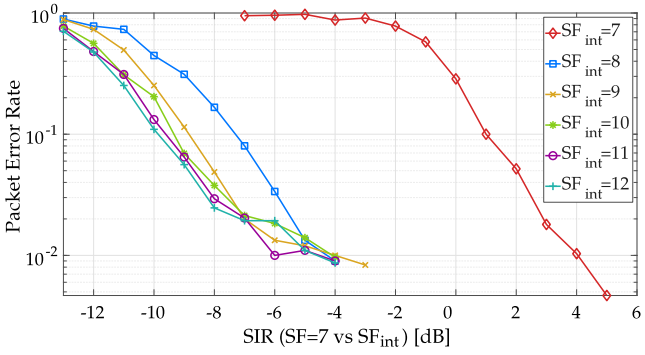


Fig. 3. PER vs SNR values in case of interference with other LoRa modulated signals.

The receiver architecture consists of three key modules: (i) the symbol detection module, responsible for identifying symbols encoded in the received signal once a new packet is correctly identified and synchronized; (ii) the synchronization module, which detects the start of a new packet and estimates the carrier and timing references utilized by the transmitter; and (iii) an optional drift tracking module to compensate for clock drifts between the transmitter and receiver, ensuring accurate demodulation of long frames.

For the transmitter architecture, besides constructing LoRa symbols, we developed a processing pipeline which includes operations such as parity check coding, whitening, shuffling, 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, the received symbol is multiplied with the synchronized base down-chirp (matching the same SF as the received signal). This results in a signal containing only two frequencies, namely $-k/T$ and $-B-k/T$, where T is the symbol time and k represents the transmitted symbol. Second, by down-sampling the signal at the rate B, both frequencies can be aliased to the same frequency $-k/T$. Finally, an FFT is applied to transform the signal into the frequency domain. The symbol index k can be estimated by analyzing the position of the peak in the output of the FFT.

An interesting feature of LoRa is the quasi-orthogonality of signals transmitted at different SFs, which contributes to its robustness against external interference sources. When the signal is multiplied with the synchronized down-chirp, an interfering signal is transformed into noise across the entire frequency band of the signal. This prevents the correct identification of the symbol peak only for very low SIR values. However, if the interfering signal is another LoRa signal with the same SF, the receiver will observe multiple peaks in the FFT output: one maximum peak corresponding to the reference symbol, and two smaller peaks corresponding to two partially overlapping interference symbols. In such a scenario, if the reference signal is a few dB stronger than the interfering one, the symbol can be successfully detected.

Fig. 2 illustrates the receiver's performance in terms of Packet Error Rate (PER) in the presence of interference with an in-band sinusoidal (or narrow-band) signal. The reference signal consists of the same packet with a payload of 20 bytes, modulated at different SFs as indicated in the labels. The figure shows that there is approximately a 3 dB increase in co-channel rejection for a unit increase in the SF used for modulation. On the other hand, Fig. 3 quantifies the receiver's robustness against interference sources generated by other LoRa signals. It 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 the presence of interference from LoRa signals modulated under different SFs (as indicated in the labels). As expected, the effect of interference is critical for an interfering LoRa signal modulated at SF 7, i.e., the same SF of the reference one. In fact, in this specific case, 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 the effects of interference are mitigated by the quasi-orthogonality of LoRa SFs: more specifically, the PER results higher than 0.1 only when the interfering signal is about 10 dB stronger than the reference one. 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.

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 exploits the preamble structure, which includes both upchirp and downchirp transmissions. The approach involves 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 $\lfloor (t-\tau)/T \rfloor - \lfloor t/T \rfloor$, creating a square wave with values 0 and ± 1 and a duty cycle of $|\tau|/T$. As a result, the output of the mixed signal features tones at only two frequencies $\nu_1 = \mathrm{CFO} - B\tau/T$ (when $t \geq \tau$) and $\nu_1 \pm B$ (when $t < \tau$). The same mixing mechanism is applied to the last part of the preamble (composed of downchirps) by multiplying the signal with base upchirps, leading to a signal with similar structure but frequencies $\nu_2 = \mathrm{CFO} + B\tau/T$ 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 calculated as $T/B \cdot (\nu_2 - \nu_1)/2$.

To identify a new preamble, the receiver follows this procedure:

(1) Samples the received signal $r(t)$ with a sampling frequency $f_s = B \cdot \mathrm{OSF}$, where OSF represents the Over Sampling Factor, and it is a multiple of the nominal bandwidth of the signal, obtaining $r_n = r(n/f_s)$.

(2) Multiplies (mixes) a window of $N = K \cdot \mathrm{OSF}$ samples with a base downchirp, resulting in:

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

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

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

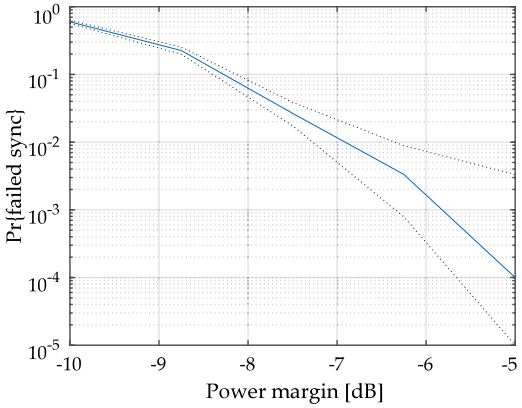


Fig. 4. Probability of failing to detect a preamble.

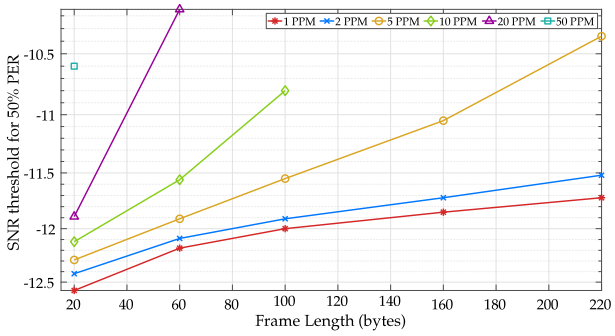


Fig. 5. SNR values guaranteeing $PER \leq 0.5$.

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

If the estimated position $\hat{k}$ is continuously detected for a certain number of times (e.g., three consecutive windows), the receiver decides that an incoming preamble is being received.

This procedure is executed continuously for each possible SF, even when the demodulation of a frame is already in progress. Remarkably, this algorithm can detect a preamble even several dBs *below* the sensitivity threshold. The probability of failing to detect a preamble is depicted in Fig. 4, where the power margin is computed as the difference between the received signal power and the receiver sensitivity.

Once the preamble is detected by checking that the sequence of estimated positions matches with the sync words, coarse estimates are obtained as previously outlined, and fine estimates of ν_1 and ν_2 can be obtained using the following equation:

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

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

4.3. Impact of clock drifts

After detecting a preamble and obtaining initial estimates of CFO and τ, the LoRa receiver needs to periodically update these estimates to compensate for clock drifts between the transmitter and receiver. Most SDR implementations do not address this issue, as they typically rely on clocks with errors in the order of 1 part per million (PPM). However, commercial devices may experience inaccuracies as high as 17 PPM due to low-cost crystal oscillators, leading to synchronization problems especially for long frames.

Fig. 5 is meant to highlight the need for a clock tracking mechanism. It quantifies the SNR threshold that ensures a Packet Error Rate (PER) lower than 0.5 for a receiver without any tracking scheme, considering different packet transmission times.

The figure illustrates the case of 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. Note that we merely choose this specific configuration as an illustrative case. 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). In other cases, packet reception is possible, but with relevant errors. Obviously, in the case of higher SFs, the effects of clock frequency errors are even worse, given the longer symbol times as compared to the case of SF 7.

A potential solution is to extend the receiver architecture with a clock tracking module, such as the one designed in [21]. This module operates as follows. After demodulating a window of consecutive symbols, the signal regenerated at the receiver is correlated with three different versions of the originally received signal: one with a time offset equal to the initial estimate τ_{est}, and two others with offsets equal to $\tau_{est} \pm 1/f_s$ (where f_s is the sampling frequency), i.e., shifted by plus or minus one sample. The correlation operations yield three different maximum values, which are then interpolated using a quadratic function. Finally, the maximum of this parabolic interpolation provides the time offset used in the current window to compensate for the clock drift. In this specific work, a time window of four symbols is chosen as a trade-off between accuracy and complexity. Note that the previous operations may be carried out approximately with the FFT-based method described in [7,8]

4.4. Transmitter implementation

To enhance robustness against synchronization errors or narrowband interference, several operations were implemented at the transmitter side, including parity check coding, whitening, shuffling, interleaving, and gray coding. These additions are particularly important for CSS-based modulations.

While most of the transmitter side operations were based on the description in the LoRa patent and application notes [6,19], certain implementation details, related to interleaving and checksum computation, required a low-level analysis of real signals transmitted by LoRa commercial devices.

To complete the transmitter design, we therefore used a commercial transmitter (namely, a Libelium device equipped with a Semtech SX1272) working with arbitrary known payloads and, at the receiver side, we searched the parameters needed to map the received symbols to the known payloads. We first identified the blocks used by the interleaver, by configuring the transmitter at the lowest 4/5 coding rate and by observing the order of the bits demodulated by our receiver. Then, we found how to compute the header checksum by transmitting frames whose length was given by a power of two, and by solving a linear system mapping the header fields into the observed checksum bits. Finally, we performed several tests regarding the CRC computation. After several trials, we found that a match between our computation and the bits transmitted by a commercial device was reached by changing the initial seed of the CRC shift register as a function of the payload length. We found these values by exhaustive search for every possible packet size.

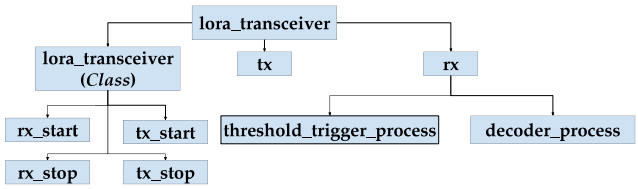


Fig. 6. Main functions of the `lora_transceiver` module of SDR-LoRa.

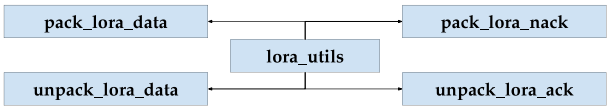


Fig. 7. Main functions of the `lora_utils` module of SDR-LoRa.

4.5. SDR LoRa: Implementation and main modules

In this section, we present a high-level description of SDR-LoRa implementation to support researchers interested in using and/or extending our open-source code. SDR-LoRa provides all the functionalities and features described in the previous sections through three main Python 3 numpy modules, namely `lora_transceiver`, `lora_utils`, and `lora`. Each module will be thoroughly described in the following.

4.5.1. Lora_transceiver

The `lora_transceiver` module directly interacts with the SDR device, manages the transmitting and receiving operations, and leverages on the `lora` module to encode and decode the complex samples of the LoRa packets. The main functions of the module are illustrated in Fig. 6, and are hereby described:

- **lora_transceiver (*Class*)**: This class allows the encapsulation of all the low-level communication aspects with the USRP SDR device. It is used to tune several configuration parameters, including receiver and transmitting frequencies for full-duplex operations, bandwidth, sample rate, and normalized signal amplitude. It also allows the set-up of the TX and RX gain of the USRP device. Among the most relevant methods of this class, there are the methods for starting and stopping the reception and transmission threads:
  - **rx_start**: This method prompts the USRP device to start the receiving operations. It also starts a receiving thread through the rx function, to handle the incoming samples and look for LoRa packets. It returns a queue object per selected Spreading Factor. The queues can be exploited by the end user to read the incoming LoRa packets.
  - **rx_stop**: This method sends a stop command to the USRP device. The latter accordingly stops recording PHY level samples from the channel.
  - **tx_start**: This method prompts the USRP device to start the transmitting operations. It also starts a transmitting thread through the tx function. Returns a queue object. Each time users puts a LoRa packet object in the queue, it is handled by the tx thread and accordingly transmitted by the USRP device.
  - **tx_stop**: If a transmitting thread is running, this method stops the thread, thus halting the transmitter operations.
- **tx**: This function handles the transmitting functionalities and directly interfaces with the USRP device. It listens for incoming packets on a queue object. Once the end users puts one or more packets in the queue, it leverages on the lora.encode function from the lora module to encode the packets into physical layer samples. Then, it fragments and sends the samples to the USRP devices.
- **rx**: This function handles the receiving functionalities and directly interfaces with the USRP device. Before starting the receiving operations, it synchronizes with an external reference clock (if the external synchronization is enabled). Then, it prompts the USRP to start recording the signal samples, and puts those samples in a buffer. The samples are then ready to be decoded. It mainly leverages on two processes:
  - **threshold_trigger_process**: This function is run as a separated process. Once a sufficient number of samples is gathered (i.e., it surpasses a SF-dependent threshold), it accordingly notifies the **decoder_process** to start decoding the samples.
  - **decoder_process**: This function is run as a separated process as well. Once the process is triggered by the **threshold_trigger_process**, it reads the signal samples available in the receiving buffer. It then leverages on the **lora.decode** function from the `lora` module to read the incoming lora packets. Note that the available samples are read through overlapping sample windows: this prevents the receiver from losing packets present on the windows overlapping zone.

4.5.2. Lora_utils

The `lora_utils` module implements several useful features, such as the encapsulation of a given data in one or more LoRa packets, the de-encapsulation of information from a sequence of packets, and also offers support for ACK/NACK functionalities. Fig. 7 shows the main functions of the module. More specifically:

- **pack_lora_data**: Given a numpy array of bytes (uint8), returns one or more LoRa packets. According to the maximum allowed packet size, the array is distributed among the packets payload. Supports the extended sequence number option, where the first payload byte is used to extend the sequence number beyond 255.
- **unpack_lora_data**: Reverses the operations performed by **pack_lora_data**. Given a numbered sequence of LoRa packets, reconstructs the original numpy array of bytes.
- **pack_lora_nack**: Given a sequence of missing LoRa packets (ARQ setting), return one or more LoRa packets whose payload is the sequence of lost packets. The output packets can be distinguished from regular data packets thank to the usage of special ID codes (**ACK_CODE** and **NACK_CODE**).
- **unpack_lora_nack**: Reverses the operations performed by **pack_lora_nack**. Given a sequence of LoRa NACK packets, reconstructs the original sequence of lost packets.

4.5.3. Lora

The `lora` module implements all the main low-level functionalities of the LoRa physical layer, including complex samples encoding and decoding, preamble detection, chirp and preamble generation, whitening, and so on. The main functionalities and features of the module (as shown in Fig. 8) are:

- **decode**: This function exposes the LoRa decoding features of the module. Its functionalities are implemented through a chain of support functions. Starting from the top:
  1. **samples_decoding**: This function returns a numpy array containing the LoRa packets decoded by the **rf_decode** function.

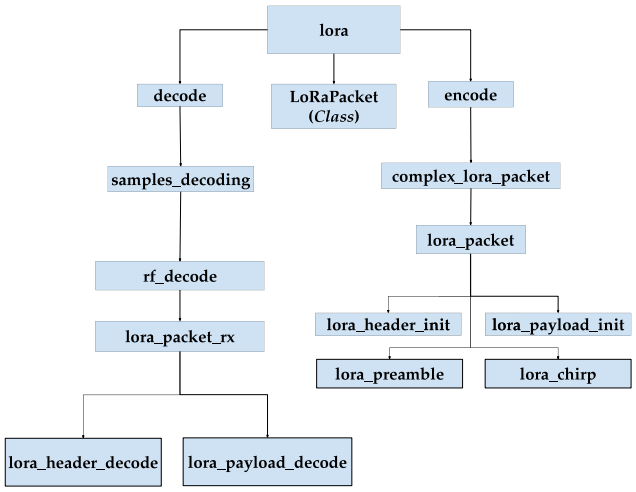


Fig. 8. Main functions of the `lora` module of SDR-LoRa.

2. **rf_decode**: This function analyzes the signal samples and finds the LoRa preamble(s), if present. It also calculates physical level parameters, such as the RSSI, the CFO error, and the preamble offset within the provided signal samples. If a preamble is present, it relies on the **lora_packet_rx** function for further analysis of the packet.
3. **lora_packet_rx**: This support function decodes (i) the packet header and (ii) the packet payload. The header and payload are decoded thanks to the auxiliary functions **lora_header_decode** and **lora_payload_decode**, respectively.

- **encode**: This function exposes the LoRa encoding features of the module. Similarly to **decode**, it leverages on a chain of support functions. Starting, once again, from the top:

1. **complex_lora_packet**: This is a support function to encode a LoRa packet. Upconverts the baseband samples returned by **lora_packet**.
2. **lora_packet**: This function builds the LoRa packet samples by leveraging on several support functions, including bit to symbol mapping, and header/payload encoding through CSS modulation. It exploits the support of several auxiliary functions, such as **lora_header_init** to initialize the header samples, **lora_payload_init** to initialize the payload samples, **lora_preamble** to generate the samples of a typical LoRa preamble, and **lora_chirp** to generate the samples of a base LoRa chirp.

- **LoRaPacket** (*Class*): This class is employed to conveniently encapsulate a LoRa packet. Includes useful information, e.g., the node SF, the packet payload, the Time of Arrival, source and destination ID, and more.

5. Applications of SDR-LoRa

We now describe three exemplary applications of SDR-LoRa which exploit some important aspects of our software-defined receiver: (i) the possibility to access useful low-level information about the physical layer; this information can be conveniently exploited to implement specific use cases, e.g., fine-grained localization; (ii) the possibility to enhance the receiver functionalities to improve the performance of standard receivers; (iii) the possibility to implement customized MAC-layer solutions on top of the SDR-LoRa physical interface.

5.1. Exploiting physical layer information for localization

Low-level access to the receiver synchronization block allows the retrieval of timing information about the packet reception times at the gateway, with a microseconds-level resolution (approximately half of the inverse of the bandwidth).

In order to maximize coverage, LoRaWAN gateways (GWs) are assumed to be mounted on rooftops. This allows a gateway to feature a GPS/GNSS receiver for network synchronization. The time-of-arrival (ToA) of every received frame can be fed to a time-difference of arrival (TDoA) -based localization algorithm to accurately localize the end devices. We implemented a simple localization scheme based on the reception times provided by SDR-LoRa.

We assumed that multiple GWs are able to receive the same frame, thus making available several independent measurements. We then aggregated the measurements collected by the independent GWs through a Least Square Localization scheme based on TDoA.

The position of the transmitter node can be estimated following an approach similar to the one described in [22]. Let us assume that a packet sent by a specific target node is received by N different GWs (with $N \geq 4$) placed at known locations, and that the strongest $RSSI$ is measured by the rth GW at the unknown distance d_r. Accordingly, we can derive the time differences $\tau_{i,r}$ between the time-of-arrival at the ith GW and the one at the rth GW, taken as reference. Indeed, the distance d_i between the target node and the ith GW can be written as $d_r + c\tau_{i,r}$, where c is the speed of the light. Let x_i and y_i denote the coordinates of a generic GW i, and x_T and y_T the coordinates of the target. The difference between the squares of the distance d_i and of the reference distance d_r can be written as:

$$(x_i - x_T)^2 + (y_i - y_T)^2 - (x_r - x_T)^2 - (y_r - y_T)^2 = 2d_r c\tau_{i,r} + c^2\tau_{i,r}^2$$

Since the quadratic terms in x_T and y_T cancel out, the equation can be rewritten as a linear equation dependent on the unknown column vector $x = [x_T, y_T, d_r]^T$. Given N gateways, it is possible to build a matrix A with $N-1$ rows, where the ith row is given by $-2[(x_i - x_r), (y_i - y_r), c\tau_{i,r}]$, and a vector b with $N-1$ components, each one equal to $[c^2\tau_{i,r}^2 - x_i^2 - y_i^2 + x_r^2 + y_r^2]$. The unknown column vector x is a solution of $Ax = b$, and a Least Square Error solution can be found as $(A^T A)^{-1} A^T b$.

5.2. Supporting interference cancellation

Another perspective application of a software-defined LoRa receiver is the extension of the receiver architecture with more advanced functionalities. In particular, we implemented a support for multi-user detection based on successive interference cancellation. The idea is (i) to generate a synthetic software version of the entire demodulated signal and (ii) to subtract this signal from the received one. In such a way, it is possible to decode other overlapping signals received at lower RSSIs. Since signal regeneration is based on the local oscillator of the receiver node, it is necessary to estimate the time and frequency jitters due to the transmitter oscillator before performing signal cancellation.

We implemented a simple solution, based on the correlation of the locally regenerated signal and the received signal in consecutive temporal windows. For each time window, we regenerate the signal with three timing offsets given by the current timing offset τ_{est} and $\tau_{est} \pm 1/f_s$. An estimate of the timing drift is obtained by looking for the maximum in the cross-correlation modulus between the incoming signal and these reconstructed signals. An even finer estimate can be obtained as the maximum of the quadratic interpolating polynomial around the aforementioned cross-correlation maximum. The sequence of estimated timing drifts is then interpolated to derive an estimation of the timing drift process, while the reconstructed signal is obtained by further interpolation of the temporary reconstructed signals. Finally, the signal cancellation can be performed by calculation the projection error of the received signal over the reconstructed one. Note that, from a complexity point of view, the implementation of this receiver

is dominated by the computational complexity of FFTs used for time correlations and of the required linear interpolations. While a sophisticated approach is certainly possible, it is worth remarking that our goal is not the design of an optimal interference-cancellation scheme, but, instead, to demonstrate that other receiver blocks and functionalities can be easily integrated into the SDR-LoRa design.

5.3. Improving link reliability

LoRaWAN networks utilize an Aloha-like random access method, which can significantly falter under heavy load situations. To enhance the dependability of connections, two strategies can be contemplated: (i) implementing an Automatic Repeat Request (ARQ) protocol to selectively retransmit corrupted frames when needed, and (ii) proactively sending supplementary frames generated using frame-level rate-less coding techniques. Both of these solutions have been integrated into the SDR-LoRa framework.

ARQ Mechanism. We integrated an optional Automatic Repeat Request (ARQ) reliability mechanism into the LoRa protocol, enhancing and complementing the inherent FEC Hamming techniques it already employs. Our ARQ protocol encompasses several sequential steps: (i) Transmitting the actual data, (ii) Receiving a response from the receiver in the form of a NACK message, (iii) Re-transmitting any missing packets as required, (iv) Sending a conclusive ACK from the receiver, and (v) Closing the communication phase. To prevent potential deadlock, if no response is received from the transmitter within a specified timeout during step (ii), the receiver takes the initiative to re-transmit the NACK packet.

Rateless Coding. Rateless codes, as described in RFC6330, operate by harnessing the potential to generate a variable number of encoded frames from a burst of k frames on the transmitting source. This coding scheme enables the receiver to decode an exact replica of the complete k-frame burst from any subset of $k+\nu$ successfully received (i.e., non-erased) encoded frames. The resulting reception overhead, denoted as ϵ and calculated as ν/k, typically remains within a few percentage points. An intriguing aspect of this solution is its independence from a downlink channel for transmitter feedback. Given the constrained downlink bandwidth prevalent in LoRaWAN networks, this approach holds practical significance in various scenarios. We introduced a signaling mechanism from the network server to devices, enabling the notification of device-specific Packet Delivery Rate (PDR). Based on this value and a desired success probability γ for delivering a k-frame burst, each participating device in the data transmission computes the quantity k' of coded frames to be generated from each group of k frames, expressed as:

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

6. Experimental results

We verified the functionality of our LoRa transceiver implementation by conducting compatibility tests between SDR LoRa and commercial LoRa devices. We set one SDR platform running the LoRa transceiver in the lab at the University of Catania, and deployed the devices in outdoor locations characterized by partially obstructed links, as depicted in Fig. 9. 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.

To thoroughly assess our prototype's performance, we also integrated the LoRa transceiver into the Colosseum testbed, which boasts an array of coordinated SDR platforms and channel emulators. A notable feature of this testbed is its ability to manipulate network scenarios, encompassing channel models between nodes as well as transmission patterns of concurrent devices. This includes the synchronization

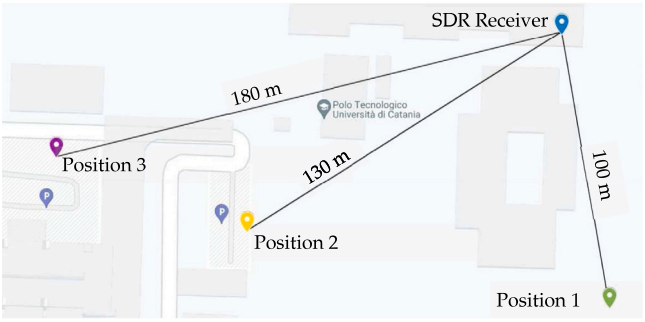


Fig. 9. Map of the testbed locations.

or the shifting of transmission attempts among independent SDR nodes, in order to explore specific interference effects stemming from multiple sources of interference. Additionally, the testbed facilitates the analysis of low-level signals captured by autonomous receivers, enabling a wide range of experimental studies on receiver architectures. For all these reasons, we chose to carry out all the experiments illustrated in the remainder of the paper over the Colosseum platform.

6.1. Testbed description

SDR-LoRa was assessed using Colosseum, which is recognized as the world's largest network emulator [23]. Colosseum operates with 128 dedicated hardware nodes, referred to as Standard Radio Nodes (SRNs), each equipped with NI/Ettus USRP X310 SDRs. These nodes can host and execute user-defined Linux Containers (LXCs), offering considerable flexibility in customizing and utilizing the underlying hardware. The connectivity among the SRN nodes is established through the Massive Channel Emulator (MCHEM), composed of several FPGA modules capable of processing radio signals using Finite Impulse Response (FIR) filters. Through this setup, the MCHEM can emulate real-world wireless RF channel effects, including attenuation, propagation delay, fading, and multipath.

Another essential component of Colosseum's architecture is the RF scenario server. A Colosseum scenario comprises wireless links between multiple SRNs, with each link defined by digital channel taps. When a scenario is activated on the emulator, these channel taps are dynamically provided to the MCHEM.

The evaluation of SDR-LoRa was conducted using a custom LXC image containing the developed SDR implementation along with the necessary libraries and system tools required for executing the code. The full LXC container has been published online, to allow reproducibility of results and experimentation with our code.[2]

6.2. SDR-LoRa Performance over Colosseum

We conducted our initial set of experiments using a single LoRa link, involving two SDR nodes fulfilling roles as transmitter and receiver. These nodes were interconnected through the SDR channel emulator. The chosen RF scenario is characterized by a noise power featuring $\sigma_n = 3.5 \times 10^{-8}$ and a tunable Path Loss.

Table 2 shows the PDR at the receiver, as a function of different channel attenuation values,[3] when the transmitter sends packets at SF 7, with a bandwidth of 125 kHz, a payload size of 50 bytes, and a fixed normalized amplitude of the signal equal to 1.

[2] To access the container, visit the URL: https://zenodo.org/record/7520494 The container password is "sdr-lora".

[3] 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.

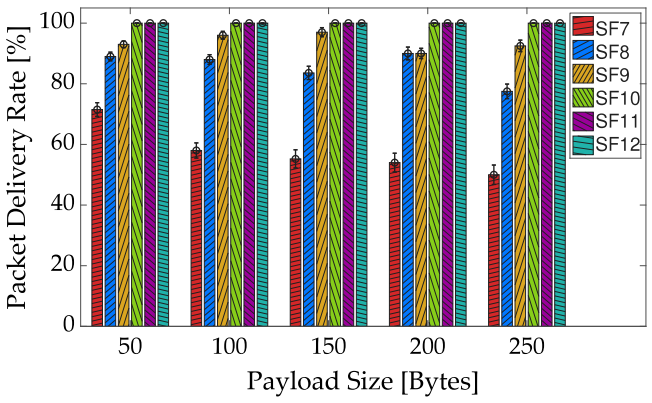


Fig. 10. PDR vs the packet Payload Size.

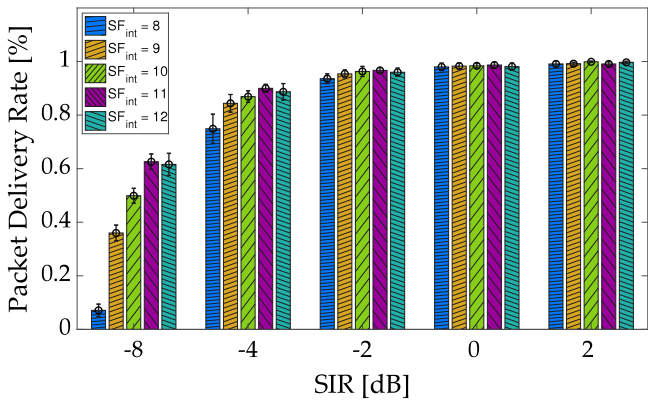


Fig. 11. PDR vs SIR for a reference link at SF 7, in presence of collisions with different SFs.

Table 2
PDR vs Channel Loss.

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

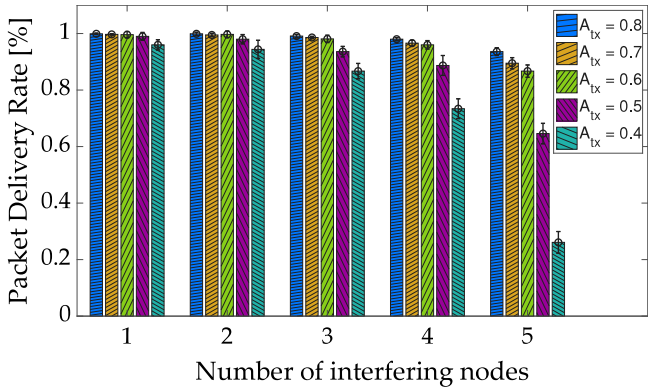


Fig. 12. PDR of a reference link at SF 7 as a function of the number of interfering nodes at SF 8.

Additionally, the link experiences supplementary power losses attributed to the connectors between SDR nodes and the wired channel emulator. Due to the complexity in quantifying these losses, we empirically determined the channel attenuation leading to an SNR below the reception threshold. In fact, the PDR results lower than 1 (approximately 75%) at a channel attenuation of 55 dB, and descends to under 15% upon increasing the attenuation to 60 dB.

For a channel attenuation setting of 56 dB (resulting in an SNR below the reception threshold for SF 7), we proceeded with additional experiments encompassing varying SFs and distinct payload sizes. Our focus was on measuring the PER while transmitting a fixed amount of data (specifically, 10000 bytes), which corresponds to a different total number of frames as a function of the employed SF. The outcomes are summarized in Fig. 10. We repeated the experiment ten times, and accordingly reported the results with a confidence interval of 99%. 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.

We executed a second set of experiments aimed at investigating the effects of interference among multiple concurrent LoRa links. Initially, we examined the inherent non-ideal orthogonality of diverse SFs. To this purpose, we established a baseline LoRa link that operated at SF 7, transmitting frames with a 50-byte payload. Concurrently, we introduced an interfering node working at a SF distinct from SF 7. This interfering node was configured for continuous transmission, without applying any duty cycle, thus ensuring that all frames transmitted by the reference link overlap with the transmissions from the interfering node.

As illustrated in Fig. 11, our experiments measured the PDR when the transmitted frames collided with interfering signals at various SFs, as a function of the SIR. We repeated the experiment ten times, and sent a total of 100 packets within each experiment. Once again, the results are reported with a confidence interval of 99%. 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 SFs 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 125 kHz, 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. 12 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 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. Similarly to the previous cases, we repeated the experiment ten times, and sent a total of 100 packets within each experiment, with a reported confidence interval of 99%.

We also investigated the effects of collisions occurring at SF 7. In theory, signals transmitted at the same SF lack orthogonality and should hinder the accurate reception of colliding frames. However, the LoRa system exhibits an exceptionally low capture threshold: it is enough a SIR of a few decibels (typically around 3 dB) in the reference link to enable proper frame demodulation. To explore this phenomenon, we replicated the experiment using multiple interfering nodes, configuring their signals at SF 7. Fig. 13 quantifies the observed

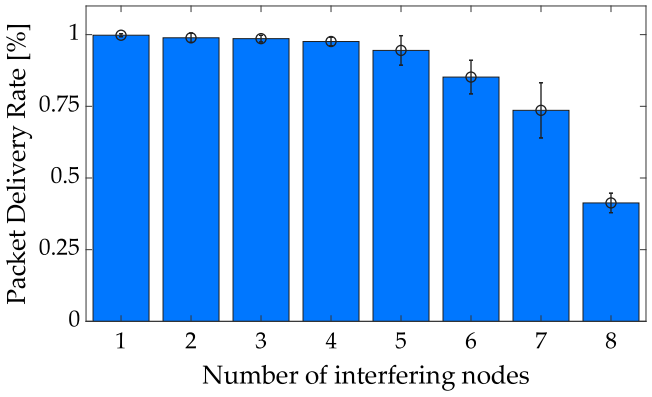


Fig. 13. PDR of a reference link at SF 7 as a function of the number of interfering nodes on the same SF.

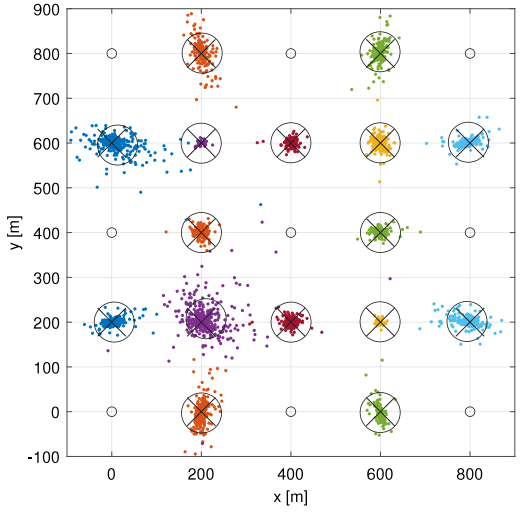


Fig. 14. Localization for $N_{GW} \geq 4$.

PDR as the number of interfering nodes increases, when the transmission power of the reference link is 6 dB stronger than the interfering ones. Once again, we repeated the experiment ten times, and sent 100 packets within each experiment. The results report a confidence interval of 99%.

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.

6.3. Using SDR-LoRa for Localization

In order to demonstrate the possibility of localizing end-devices by using physical information provided by SDR-LoRa, we built a scenario in the Colosseum testbed with 25 nodes placed in a grid, whose regular distance is set to 200 m. To ease the evaluation of this proof-of-concept, we employed a single-path channel with a free space path loss. Eight edge nodes and the node in the center act as GWs, while the other sixteen act as end-devices. End devices are configured for sending sporadic packets at a rate of one packet every 10 s. A total number of 400 packets per end-device are tracked by the central server, by collecting the reception times measured by all the GWs in radio visibility (i.e. able to correctly demodulate the packets).

A complete trace with all the reception times and relevant packet identifiers measured by each GW is then processed by the positioning

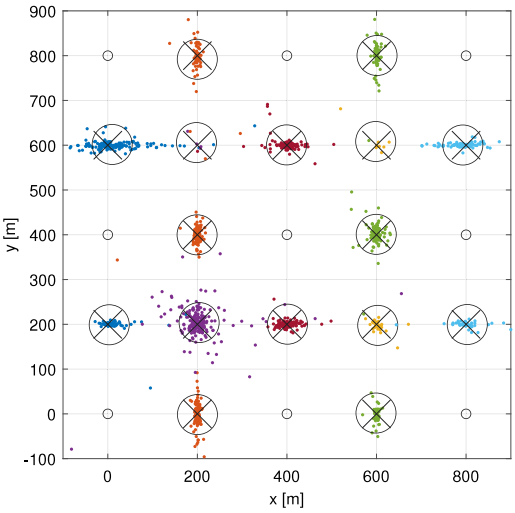


Fig. 15. Localization for $N_{GW} = 4$.

Table 3
Positioning errors in meters: mean errors along x and y, with standard deviations, and mean positioning error distance.

N_{GW}	$E[e_x]$	σ_x	$E[e_y]$	σ_y	$E[d]$
4	1.13	31.8	0.47	20.7	19.6
5	0.94	22.1	0.50	17.4	16.9
6	1.77	22.9	0.84	19.1	18.0
7	2.44	24.5	0.96	20.8	19.9
8	3.25	25.4	1.22	21.9	21.0
9	4.40	29.0	1.17	23.9	22.5

scheme. For each packet in the trace, an estimate of the transmitter position can be evaluated, and different estimates can be obtained by changing the subset of ToA measurements used for the calculations.

Fig. 14 and Fig. 15 respectively show the estimates provided by the chosen positioning scheme using all the available measurements or only the best four ones. Four measurements are the minimum required for TDoA-based positioning in 2-D with the chosen strategy. Each estimate is represented by a different colored point. The figures depict the network topology in terms of node positions (in a grid of 800 m × 800 m), distinguishing between gateways (white dots) and end nodes (crosses), as well as the average values of the estimates of the transmitter position (big circles). By comparing the two figures, it is evident that using the best GWs generally provides better results, as the measurements of the propagation delays are more reliable. Moreover, we can also observe that the positioning errors differ as a function of the end-device placement in the grid, with some evident border effects. Some statistics about the positioning errors are reported in Table 3. A close inspection of the table reveals that the localization algorithm yields the best performance for $N_{GW} = 5$. It is necessary to note that the high performance of the localization strategy can be explained by means of the considered propagation model. In this scenario, the received signal strength results several tens of dBs above the sensitivity threshold and the time of arrival estimates feature a very low standard deviation, on the order of 0.1 μs. A more realistic evaluation would involve employing, for example, an Urban Macro channel model with NLOS propagation, and the localization performance is expected to decrease accordingly, both for lower reliability estimates and for propagation delay effects.

6.4. Using SDR-LoRa for Interference Cancellation

We assessed the performance of innovative LoRa receivers built on top of SDR-LoRa by analyzing the effect of cancellation in the simplest

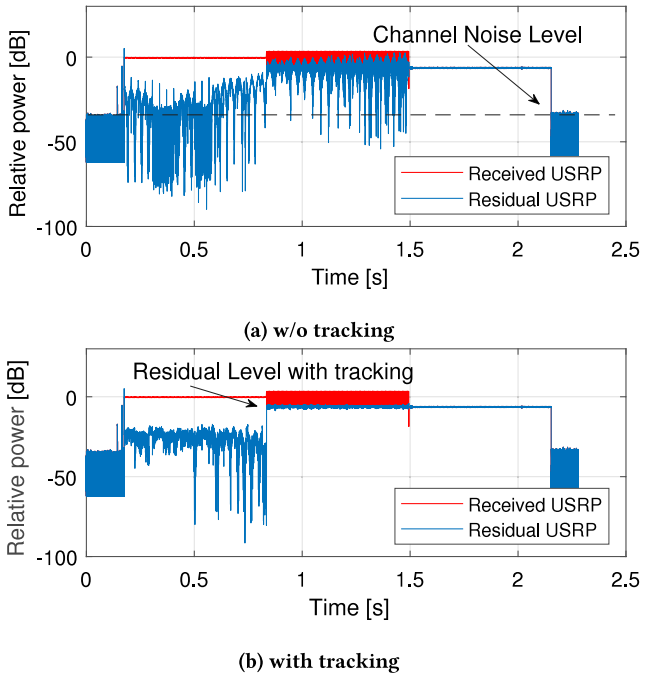


Fig. 16. Synchronization and Interference Cancellation example.

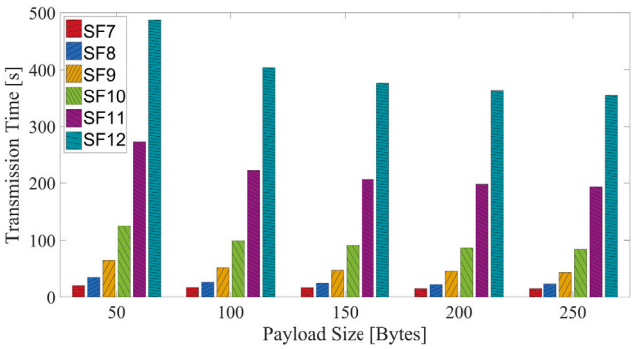


Fig. 17. Transmission vs the packet Payload Size.

Table 4
Overheads of Raptor-Q coding.

SF	100 bytes			250 bytes		
	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

possible collision case, i.e., between two partially overlapping frames. We measured the residual signal (that is, the difference between the received signal and the regenerated reference frame) in two different type of experiments, where the signal is regenerated, with and without the clock drifting tracking.

Fig. 16 from [21] compares the results obtained on a sample collision in the two types of experiments. In particular, the figures refer to a collision between two frames modulated at SF 12 and demodulated by our SDR receiver before (received signal, in red) and after cancellation of the first received frame (residual signal, in blue). SF12 represents a worst-case scenario, as the effects of clock drift are more severe with longer frames (and, as remarked in Section 3.2, larger SFs yield longer frames).

Note that in both the experiments channel noise and clock drift affect the demodulation operations. In particular, Fig. 16(a) shows the results of the interference cancellation scheme without applying the tracking algorithm, while Fig. 16(b) shows the same when applying also the clock tracking scheme. From the two figures, it is evident that the residual noise is more pronounced in the former case. In fact, it turns out that the variance of the squared absolute value of the residual signal with no clock tracking is about 0.0109, while in the other case the variance drops to $8.5267 \cdot 10^{-5}$. Moreover, note that USRP clocks are more precise than those found in commercial devices (datasheets report ±2.0ppm for a USRP with TCXO versus ±17ppm found in commercial LoRa modules datasheets with a simple XO, i.e., almost an order of magnitude higher). Accordingly, the impact of clock drifting is more relevant when implementing a cancellation algorithm with commercial end devices, reinforcing the need of the proposed tracking mechanisms.

6.5. Using SDR-LoRa at Data Link

Finally, we also tested the link layer schemes implemented on top of SDR-LoRa to improve the link reliability. Fig. 17 quantifies the total amount of time needed to successfully transfer 10000 bytes of data to the receiver, for a channel attenuation of 56 dB. The calculation of the transmission time takes into account the total interval required for transmitting all the frames, including selective retransmissions of corrupted ones. 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, especially in presence of data bursts, is exploiting rate-less coding. Table 4 summarizes the number of coded frames required for guaranteeing a delivery probability $\gamma \leq 1 - 10^4$, as well as 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, according to RFC6330, guarantees 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 greater than the one resulting from selective retransmissions, but the difference gets lower as 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).

This idea is better quantified in Fig. 18, where we plot the overhead with respect to the minimum number of encoded frames as derived by the erasure channel capacity, i.e. k/PDR. The figure shows how the required overhead becomes negligible as the number of frames in a burst increases.

7. Conclusions and future works

In this paper we described SDR-LoRa, an open-source, full-fledged implementation of a LoRa transceiver for SDR platforms. Moreover, in order to demonstrate the flexibility of our implementation, we built some innovative applications built on top of SDR-LoRa, namely fine-grained localization, interference cancellation, and enhanced link reliability. The SDR-LoRa prototype has been fully integrated into the Colosseum testbed and has been publicly released to enable access to low-level LoRa signals, thus fostering advanced research on the topic.

We extensively evaluated our prototype in both real life, and Colosseum-based experiments. Differently from previous studies, where several virtual end devices were usually emulated by means of a single SDR platform, the usage of the Colosseum testbed allows the configuration and control of multiple and independent end devices and gateways. First, we demonstrated the compatibility of SDR-LoRa

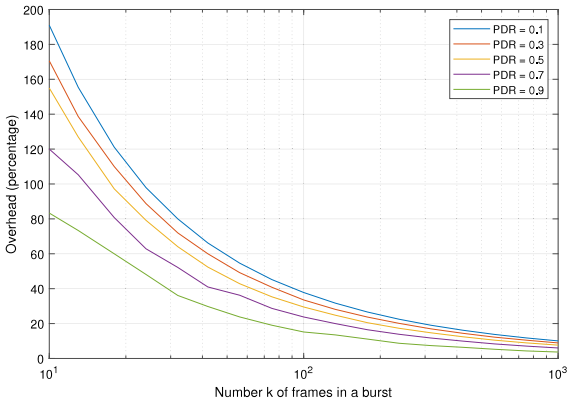


Fig. 18. Overhead of RFC6330 rateless coding with respect to the erasure channel capacity limit, for a burst of k frames and several PDR values.

with real-world commercial devices. Then, we described and evaluated the three aforementioned applications of SDR-LoRa. Among the main insights of the experiments: (i) SDR-LoRa offers a great performance in the localization of the end devices. In fact, as shown in Section 6.3, SDR-LoRa is able to estimate the position of a LoRa node with a mean error as low as about 17 m, over a grid of 800 m × 800 m; (ii) SDR-LoRa is able to significantly improve the decoding of overlapping frames through successive interference cancellation. This feature unlocks several new possibilities, such as the usage of multiple gateways to decode interfering signals in densely populated LoRa networks; (iii) SDR-LoRa allows to improve and extend the built-in link reliability schemes implemented in LoRaWAN. In particular, the experiments carried on in Section 6.5 demonstrated how ARQ schemes offer a good performance, thereby enabling lossless applications over the LoRa protocol.

Future works may involve, but not be limited to:

- improvement of the tracking of time and frequency drifts. This improvement is fundamental in the case of low quality local oscillators and/or high mobility of end nodes or gateways (e.g. the satellite LoRa scenario), where the decoding of long LoRa frames (i.e., with a large payload and/or a high SF) would be otherwise impossible;
- improved time-of-arrival estimates to aid localization, possibly powered by the above mentioned improved synchronization strategies. Improving the estimates is crucial to perform fine-grained localization, e.g., for indoor environments;
- improved interference cancellation strategy, where unsupervised machine learning techniques may be involved in order to cope with hard-to-model variations in the amplitude of the envelope of the received signals. In such a way, it would be to decode multiple signals even in the case of massive interference, e.g., for completely overlapping signals or for signals way below the Capture threshold.

CRediT authorship contribution statement

Fabio Busacca: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Stefano Mangione:** Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Sergio Palazzo:** Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Francesco Restuccia:** Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Ilenia Tinnirello:** Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing.

Declaration of competing interest

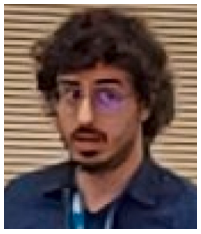
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Source code available on GitHub.

References

[1] J.P.S. Sundaram, et al., A survey on LoRa networking: Research problems, current solutions, and open issues, IEEE Commun. Surv. Tutor. 22 (1) (2019) 371–388.
[2] A. Ikpehai, et al., Low-power wide area network technologies for internet-of-things: A comparative review, IEEE Internet Things J. 6 (2) (2018) 2225–2240.
[3] U. Raza, et al., Low Power Wide Area networks: An overview, IEEE Commun. Surv. Tutor. 19 (2) (2017) 855–873.
[4] A. Al-Shawabka, et al., DeepLoRa: Fingerprinting LoRa devices at scale through deep learning and data augmentation, in: Proceedings of ACM Mobihoc 2021, 2021, pp. 251–260.
[5] D. Garlisi, et al., Interference cancellation for LoRa gateways and impact on network capacity, IEEE Access 9 (2021) 128133–128146.
[6] O.B.A. Seller, N. Sornin, Low power long range transmitter, 2014, European Patent EP2763321A1.
[7] O.B.A. Seller, N. Sornin, Low complexity, low power and long range radio receiver, 2018, European Patent EP3264622B1.
[8] C. Bernier, et al., Low complexity LoRa frame synchronization for ultra-low power software-defined radios, IEEE Trans. Commun. 68 (5) (2020) 3140–3152.
[9] M. Knight, et al., Decoding LoRa: Realizing a modern LPWAN with SDR, in: Proceedings of the GNU Radio Conference, vol. 1, (no. 1) 2016.
[10] A. Marquet, et al., Towards an SDR implementation of LoRa: Reverse-engineering, demodulation strategies and assessment over Rayleigh channel, Comput. Commun. 153 (2020) 595–605.
[11] P. Robyns, et al., gr-lora: An efficient LoRa decoder for GNU Radio, 2017, https://github.com/rpp0/gr-lora.
[12] J. Tapparel, et al., An open-source LoRa physical layer prototype on GNU radio, in: 2020 IEEE SPAWC, 2020, pp. 1–5.
[13] Z. Xu, S. Tong, P. Xie, J. Wang, From demodulation to decoding: Toward complete LoRa PHY understanding and implementation, ACM Trans. Sensor Netw. 18 (4) (2023) 1–27.
[14] F. Busacca, S. Mangione, I. Tinnirello, S. Palazzo, F. Restuccia, SDR-LoRa: Dissecting and implementing LoRa on software-defined radios to advance experimental IoT research, in: Proceedings of the 16th ACM Workshop on Wireless Network Testbeds, Experimental Evaluation & CHaracterization, 2022, pp. 24–31.
[15] Semtech Corporation, LoRa® and LoRaWAN®: A technical overview, 2020.
[16] Semtech Corporation, LoRa™ Modulation Basics, 2015.
[17] L. Vangelista, Frequency shift chirp modulation: The LoRa modulation, IEEE Signal Process. Lett. 24 (12) (2017) 1818–1821.
[18] M. Chiani, A. Elzanaty, On the LoRa modulation for IoT: Waveform properties and spectral analysis, IEEE Internet Things J. 6 (5) (2019) 8463–8470.
[19] Semtech, SX1272/3/6/7/8: LoRa modem - designer's guide, 2013.
[20] Semtech Corporation, SX1272/73 - 860 MHz to 1020 MHz low power long range transceiver datasheet. Available online: https://www.semtech.com/products/wireless-rf/lora-connect/sx1272.
[21] D. Garlisi, et al., Interference cancellation for LoRa gateways and impact on network capacity, IEEE Access 9 (2021) 128133–128146.
[22] S.M. Sheikh, H.M. Asif, K. Raahemifar, F. Al-Turjman, Time difference of arrival based indoor positioning system using visible light communication, IEEE Access 9 (2021) 52113–52124.
[23] L. Bonati, et al., Colosseum: Large-scale wireless experimentation through hardware-in-the-loop network emulation, in: 2021 IEEE DySPAN, pp. 105–113.



Fabio Busacca received the Ph.D. degree in Information and Communication Technologies from the University of Palermo, Italy, in 2022. He is currently an assistant professor at the University of Catania, Italy. His main research interests are LPWAN protocols for the IoT, artificial intelligence and game theory applied to next-generation communication networks, and underwater acoustic communications.

Stefano Mangione received the degree (cum laude) in electronics engineering, telecommunications curriculum, in 2000. He is currently an Assistant Professor of telecommunications engineering with the University of Palermo. His research activities have been focused on physical layer aspects of communication systems, from forward error correction coding to equalization strategies for spread spectrum systems. Recently his activity includes automatic methods for registration of nuclear magnetic resonance images, multiuser receiver strategies for LoRa, and the study of underwater communication systems.

Sergio Palazzo received the degree in electrical engineering from the University of Catania, Catania, Italy, in 1977. Since 1987, he has been with the University of Catania, where he is currently a Professor of Telecommunications Networks. In 1994, he spent the summer with the International Computer Science Institute, Berkeley, as a Senior Visitor. In 2003, he was with the University of Canterbury, Christchurch, New Zealand, as a recipient of the Visiting Erskine Fellowship. His current research interests include wireless networks, mobile systems, intelligent techniques in network control, multimedia traffic modeling, and protocols for the next generation of the Internet. Prof. Palazzo has been serving on the Technical Program Committee of INFOCOM, the IEEE Conference on Computer Communications, since 1992. He has been the General Chair of some top-level conferences, including ACM MobiHoc 2006, ACM MobiOpp 2010, ACM MobiHoc 2019, and IFIP Networking 2022, and currently is a member of the MobiHoc Steering Committee. He has also been the TPC Co-Chair of some other conferences, including IFIP Networking 2011, IWCMC 2013, and European Wireless 2014. He also served on the Editorial Board of several journals, including IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE Wireless Communications Magazine, Computer Networks, Ad Hoc Networks, and Wireless Communications and Mobile Computing.

Francesco Restuccia received the Ph.D. degree in computer science from the Missouri University of Science and Technology, Rolla, MO, USA, in 2016. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, and a Member of the Institute for the Wireless Internet of Things and the Roux Institute, Northeastern University, Boston, MA, USA. He has authored or coauthored more than 60 papers in top-tier venues in computer networking, as well as coauthored 16 U.S. patents and three book chapters. His research focuses on the design and experimental evaluation of next-generation edge-assisted data-driven wireless systems. Dr. Restuccia's research is funded by several grants from the U.S. National Science Foundation and the Department of Defense. He was the recipient of the Office of Naval Research Young Investigator Award, the Air Force Office of Scientific Research Young Investigator Award, and the Mario Gerla Award in Computer Science, as well as Best Paper Awards at IEEE INFOCOM and IEEE WOWMOM. He regularly serves as a TPC Member and Reviewer for several ACM and IEEE conferences and journals.

Ilenia Tinnirello is a Full Professor at the University of Palermo, Italy. She has been Visiting Researcher at the Seoul National University, Korea, and the Nanyang Technological University of Singapore. Her research activities focus on wireless networks, and in particular on the design and prototyping of protocols and architectures for emerging reconfigurable wireless networks. She has been involved in several European research projects.