Collision Resolution Protocol for Delay and Energy Efficient LoRa Networks

Nancy El Rachkidy, Alexandre Guitton, Megumi Kaneko

SS5G: Collision Resolution Protocol for Delay and Energy Efficient LoRa Networks

Nancy El Rachkidy[(1)], Alexandre Guitton[(1)], Megumi Kaneko[(2)]

(1) Université Clermont Auvergne, CNRS, LIMOS, F-63000 Clermont-Ferrand, France

(2) National Institute of Informatics, Hitotsubashi, 2-1-2, Chiyoda-ku, 101-8430 Tokyo, Japan

Emails: nancy.el_rachkidy@uca.fr, alexandre.guitton@uca.fr, megkaneko@nii.ac.jp

Abstract

Future 5G and Internet of Things (IoT) applications will heavily rely on long-range communication technologies such as low-power wireless area networks (LPWANs). In particular, LoRaWAN built on LoRa physical layer is gathering increasing interests, both from academia and industries, for enabling low-cost energy efficient IoT wireless sensor networks for, e.g., environmental monitoring over wide areas. While its communication range may go up to 20 kilometers, the achievable bit rates in LoRaWAN are limited to a few kilobits per second. In the event of collisions, the perceived rate is further reduced due to packet loss and retransmissions. Firstly, to alleviate the harmful impacts of collisions, we propose a decoding algorithm that enables to resolve several superposed LoRa signals. Our proposed method exploits the slight desynchronization of superposed signals and specific features of LoRa physical layer. Secondly, we design a full MAC protocol enabling collision resolution. The simulation results demonstrate that the proposed method outperforms conventional LoRaWAN jointly in terms of system throughput, energy efficiency as well as delay. These results show that our scheme is well suited for 5G and IoT systems, as one of their major goals is to provide the best trade-off among these performance objectives.

Index Terms

LoRa, LoRaWAN, LPWAN, Collision Resolution, Interference Cancellation, Desynchronized Signals.

I. Introduction

Long-range low-power communication technologies such as LoRa [1], Sigfox [2], and Ingenu [3], are becoming widely used in Low-Power Wide Area Networks (LPWANs) [4], [5], [6]. These technologies enable to cover extensive zones with very low energy consumption and are thus attractive technologies for supporting the future Internet of Things (IoT) communications and applications, in particular environmental monitoring [7], [8], [9].

LoRa [1] is a recent physical layer for LPWANs making use of Chirp Spread Spectrum (CSS) modulations, which can adaptively extend the communication range by reducing the achievable throughput. On top of this LoRa physical layer, LoRaWAN [10] defines a simple MAC protocol based on open specification, which allows end-devices to communicate to a network server through gateways, but with a small duty-cycle (e.g., 1%). Thus, end-devices can save energy, and the network lifetime is increased. The main issue in LoRa and LoRaWAN is their throughput limitation: the indicative physical bitrate varies between 250 and 11000 bps [11]. Moreover, when two end-devices transmit simultaneously using the same parameters such as channel, Spreading Factor (SF), and are received by the gateway with a similar power, a collision occurs and none of the signals are decoded by LoRa. Thus, both end-devices have to retransmit, which further reduces their achievable throughput.

So far, a number of works have focused on channel and SF allocation issues for the uplink transmissions of LoRa systems, among which [12], [13], [14]. Most of these methods rely on a centralized scheduling unit at the gateway. The feasibility of large-scale LoRa networks has been analyzed in [15], [16], in particular the effect of co-SF interferences as a large number of end-devices may use the same SF at the same time. Most previous works consider SFs to be orthogonal, but recently, various experiments and analysis have pointed out the impact of imperfect orthogonality of SFs whereby devices using different SFs may interfere among themselves [17], [18], [19].

To alleviate the large performance degradations due to co-SF interferences, we have proposed in [20] a method for decoding superposed LoRa signals by exploiting the specific features of LoRa signals. The proposed algorithm was shown to provide significant performance enhancements in terms of achievable throughput, for different SF levels. However, the algorithm in [20] was solely designed to handle two superposed LoRa signals and we did not consider any MAC protocol.

Therefore, in this work, we extend our preliminary proposal of [20] by designing a general decoding algorithm for several signals, which is far more intricate than the restrictive case

of two superposed signals. In addition, we propose a tailored MAC protocol on top of our decoding algorithm. In particular, we show that it is possible to retrieve the frames from superposed signals that are slightly desynchronized, with reasonable assumptions on the hardware.

Our contributions are three-fold. Firstly, we propose an algorithm that is able to cancel the collision between two collided signals and thus retrieve entire frames without any loss. We then generalize this algorithm for retrieving several collided frames that are sent by several end-devices. Secondly, we propose a MAC layer slotted with beacons in order to allow synchronized transmissions (and to compensate for the drifting of the end-devices). This MAC layer divides time into slots in which several end-devices might send slightly desynchronized frames. Thirdly, we propose a Cyclic Redundancy Check (CRC) decoding scheme that can be applied in order to decode the few frames that our algorithm was unable to decode.

The structure of this paper is as follows. Section II describes the LoRaWAN technology with the LoRa physical layer and the LoRaWAN MAC layer. Section III presents the proposed decoding algorithm designed to correctly decode the collided frames, followed by the proposed MAC layer in Section IV. Section V shows the simulation parameters we use and the results we obtained. Finally, Section VI concludes the paper.

II. LoRaWAN Description

In the following, we first describe the LoRa physical layer, which is the main focus of our paper. Then, we describe the LoRaWAN MAC protocol.

A. LoRa

LoRa [1] is a physical layer technology for LPWAN, based on a Chirp-Spread Spectrum (CSS) modulation. Each LoRa chirp consists of a linear frequency sweep. The duration of the sweep is called symbol duration (SD), and depends on the value of the spreading factor SF and on the bandwidth BW. The sweep is performed over the whole bandwidth BW. Chirps are either up-chirps, where the frequency sweep is increasing, and down-chirps, where the frequency sweep is decreasing.

Each chirp is a symbol and can encode 2^{SF} possible values. This is achieved by shifting the sweep by the symbol value, as shown on Figure 1 for an up-chirp. From the sharp edge in the instantaneous frequency trajectory [21], and assuming that the receiver is synchronized with the transmitter, the receiver can compute the symbol value as the shift in the frequency

at the beginning of the symbol. The symbol value of an up-chirp is also proportional to the remaining time between the sharp frequency edge and the end of the symbol, as shown on Figure 1. The symbol value of a down-chirp is proportional to the time between the beginning of the symbol and the sharp frequency edge.

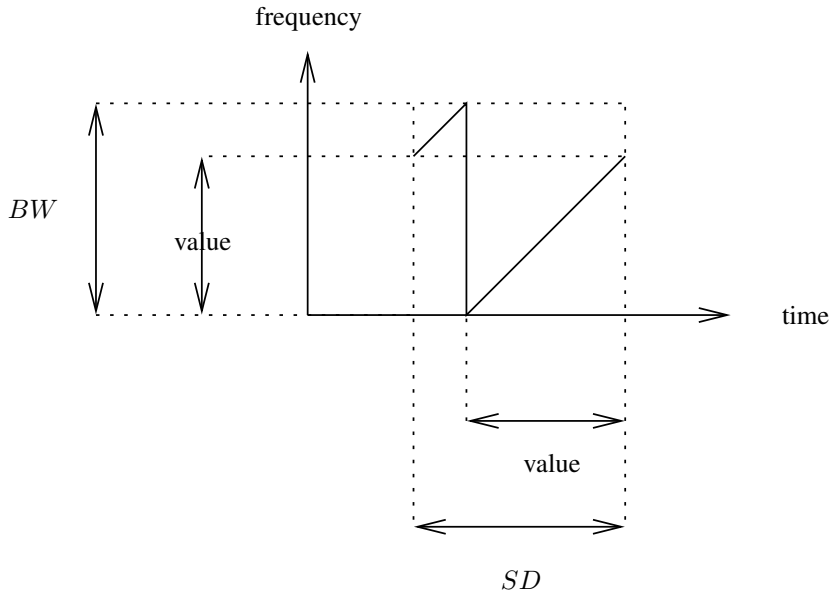

Figure 1. Example of a single LoRa up-chirp. Computing the symbol value requires knowing the symbol start time and the initial frequency, or the sharp frequency edge and the symbol end time.

To decode a symbol, the receiver needs to know the frontier of the symbol. Thus, LoRa synchronizes the transmitter and the receiver by using a preamble of a few symbols. In the case of uplink communications, the preamble consists of three parts: (i) a series of up-chirps (generally six), each having a symbol value of 0, (ii) two up-chirps encoding the sync word, which is a network identification, and (iii) two and a quarter down-chirps, used to identify the end of the preamble. The payload and a CRC follow the preamble, and are encoded using up-chirps. LoRa allows an explicit header mode, which inserts a header between the preamble and the payload. This header contains the payload length, the coding rate, and an optional header CRC.

Figure 2 shows an example of an uplink communication with a shorten preamble (two up-chirps instead of six, no sync word, and one down-chirp instead of two and a quarter) and a few data symbols (four symbols). We chose SF3 for the sake of simplicity, leading to $2^{SF} = 8$ possible values per symbol. Let us assume that a desynchronized node starts receiving the preamble, not necessarily at the exact beginning of the preamble. The node detects a sharp frequency edge of the preamble, which indicates the frontier of a symbol.

From this information, the receiver can synchronize itself according to the transmitter. The end of the preamble is detected by the inversion of the chirps. Then, the payload is decoded. In this example, the data symbols are 6, 0, 4, 4.

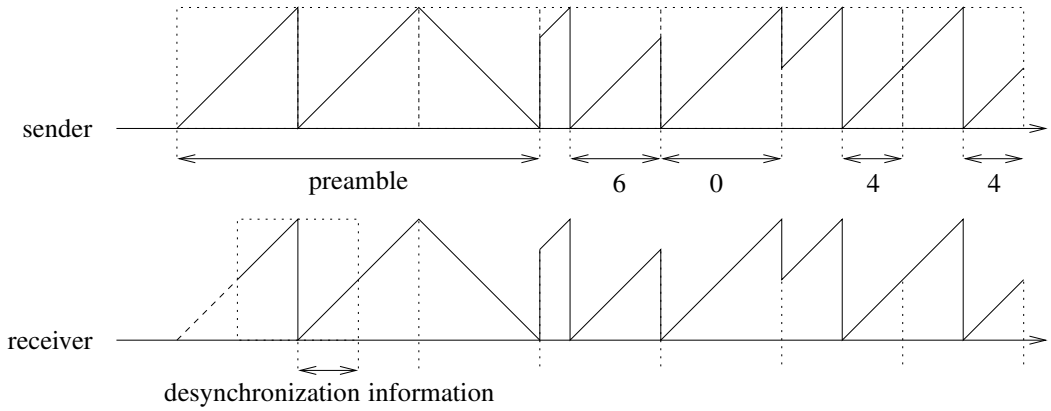

Figure 2. Example of a LoRa uplink frame, with a short preamble and four data symbols, with SF3. The receiver synchronizes itself with the sender during the preamble.

B. LoRaWAN

LoRaWAN (in version 1.0 [22] or in version 1.1 [10]) is a simple MAC layer. It is based on the LoRa physical layer. The topology defined in LoRaWAN is a star topology where end-devices are connected to a network server through relays called gateways. The communication technology between the end-devices and the gateways is based on CSS modulations. Moreover, LoRaWAN defines three classes for end-devices: class A is for low-power uplink communications, class B is for delay-guaranteed downlink communications, and class C is for end-devices without energy constraints. In class A, which is the only mandatory class, the end-devices are energy-efficient. In this class, the end-devices can transmit at any time using ALOHA mechanism: an end-device chooses a channel randomly, sends the frame, and waits for an acknowledgement during two successive receive windows. The transmission time of each end-device should not exceed 1%.

LoRaWAN manages the bitrate according to the quality of links. Indeed, it uses the SF of the signal in order to have a trade-off between the robustness of the signal and the bitrate. When an end-device experiences a low signal quality, it increases its SF in order to be able to send frames over long distances and thus better decode the signal. However, this results into lower bitrate. This adaptation is controlled by the datarate (DR) of LoRaWAN, which varies from DR0 (for large SF but small bitrate) to DR6 (for small SF but larger bitrate).

The European regional settings of LoRaWAN [11] define most LoRa parameters. The bandwidth of channels, BW, is equal to 125 kHz for DR0 to DR5, and 250 kHz for DR6. SF varies from 12 down to 7 for DR0 to DR5, and is equal to 7 for DR6. The preamble length is equal to 6 symbols. The physical bitrate varies between 250 bps for DR0, to 11000 bps for DR6. The maximum MAC payload of a frame varies between 59 bytes for DR0 and 230 bytes for DR6.

III. Proposed Superposed LoRa Signal Decoding

LoRa gateways are able to decode superposed LoRa signals as long as they are sent on different channels or on different SFs. Notice however that some researchers have shown that signals on different SFs are not completely orthogonal [17], [18], [19].

When several signals are received on the same channel and with the same SF, a difference of received power might cause the strongest signal to be captured [21], [23]. When several signals have a similar receive power, a collision occurs and all signals are considered lost [15], [16].

In this paper, we focus on decoding superposed LoRa signals of *similar receive power*, on the *same channel*, with the *same SF*. To do so, we show that we can use timing information to match the correct symbols to the correct end-device.

In Subsection III-A, we describe our assumptions. In Subsection III-B, we provide our main algorithm, and we describe how it can decode two signals that are slightly desynchronized. In Subsection III-C, we extend the algorithm for the case of three or more signals that are slightly desynchronized. In Subsection III-D, we show how the CRC of frames can be used to decode additional frames.

A. Assumptions

As in [20], we assume that there are no non-linearity effects between up-chirps (respectively down-chirps). In other words, if two up-chirps (resp. down-chirps) c_1 and c_2 overlap at a given time t at the receiver side, the two observed frequencies are the frequency of c_1 (at time t) and the frequency of c_2 (at time t). Without additional information, it is not possible to correlate the frequency to the corresponding transmitter. We assume that when an up-chirp is superposed with a down-chirp, it is not possible to detect any of the frequencies. We assume that when several frequencies overlap at a given time, only one frequency is detected by the receiver. For instance, if there are three nodes transmitting at a given time, but only two frequencies f_1 and f_2 are detected, we assume that it is not possible to know whether two

nodes were transmitting with f_1 and one with f_2, or one node was transmitting with f_1 and two with f_2.

We assume that it is possible for the hardware of the receiver to detect all frequencies of overlapping up-chirps (resp. down-chirps) within δ time-units. In the following examples, we use $\delta = SD/4$ unless stated otherwise. Please note that on real LoRa hardware, the decoding of signals is not carried out by directly detecting the sharp frequency edges, but instead by computing a fast Fourier transform and detecting the peak in the frequency domain [21]. With our proposition, this translates into either detecting the two sharp frequency edges in the time domain, or the two peaks in the frequency domain. In practice, it is likely that δ cannot be too small, as uncertainties in frequency detection might occur.

We also assume some properties on the frames: all nodes transmit with the same preamble duration, the frame length is included in the explicit header, and there is at least one symbol change during the whole frame: that is, the payload (data and CRC) does not consist of a sequence of identical symbols.

Finally, we consider that nodes are slightly desynchronized: all nodes start their transmission within $SD - \delta$ time units, and during the whole transmission duration, the transmissions of any two nodes have a delay of at least δ time units. In the following examples, we assume that each node n_i starts transmitting at time $t_0 + (i-1)\delta$ (for $i \geq 1$), and we consider that time drift between transmitters is negligible as the time on air of LoRa frames is short.

B. Case of two slightly desynchronized signals

In this subsection, we consider the superposition of two signals from two transmitters that are slightly desynchronized (by at least δ time units, and at most $SD - \delta$ time units).

Figure 3 shows the superposition of two slightly desynchronized signals. The preamble length is three symbols (2 up-chirps instead of 6, no sync word, and 1 down-chirp instead of 2.25), and SF3. The figure shows the signal of the first transmitter n_1 starting at t_0, the signal of the second transmitter n_2 starting at $t_0 + \delta$, and the superposed signal at the receiver. Note that the data transmitted by n_1 is $(2, 2, 6, 4, 4)$, and the data transmitted by n_2 is $(6, 0, 4, 6, 2)$. We will first explain our algorithm on this example, and then proceed with a more formal description.

Example of preamble detection and data decoding

Preamble detection: During $[t_0; t_0 + \delta]$, the receiver detects the preamble of n_1. During $[t_0 + \delta; t_0 + 2\delta]$, the receiver is able to detect that two slightly desynchronized signals are transmitted, and is able to deduce the symbol frontiers of both transmitters. At frontier t_1,

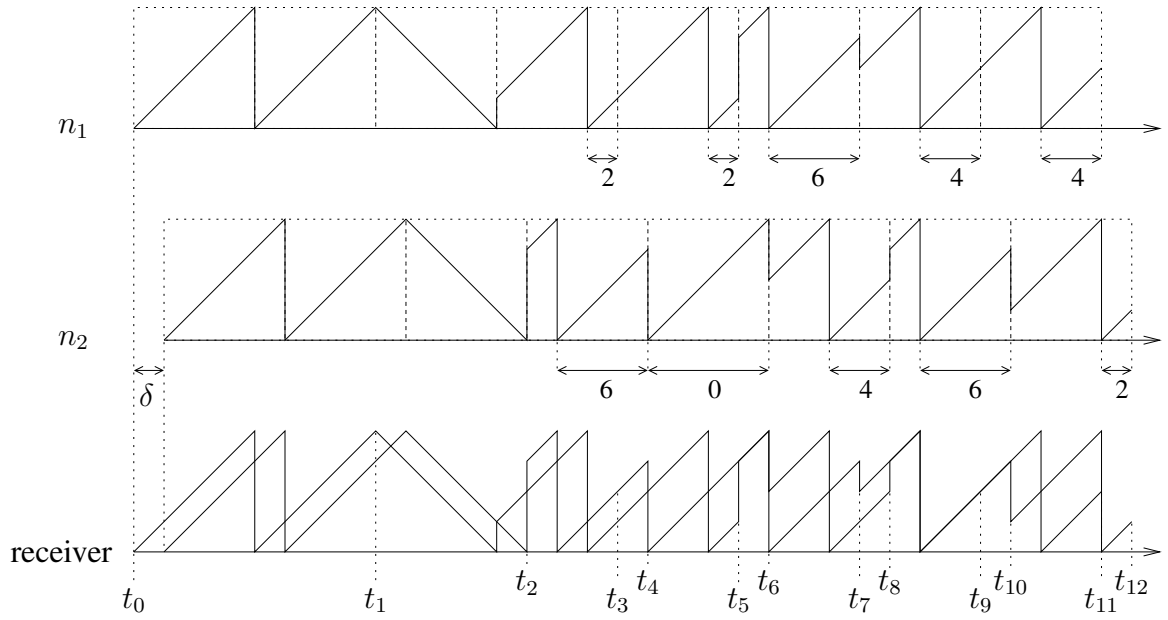

Figure 3. The superposition of two slightly desynchronized signals produces a complex signal, which can still be decoded in linear time.

time	F_-	F_+	symbol	time	F_-	F_+	symbol
t_2	unknown	$\{4,6\}$	initialization	t_3	$\{2,4\}$	$\{2,4\}$	$s_1^1 = *, s_2^1 = s_1^1$
t_4	$\{4,6\}$	$\{0,4\}$	$s_1^2 = 6, s_2^2 = 0$	t_5	$\{2,6\}$	$\{6\}$	$s_2^1 = 2, s_3^1 = 6$
t_6	$\{0\}$	$\{0,4\}$	$s_2^2 = 0, s_3^2 = 4$	t_7	$\{2,6\}$	$\{2,4\}$	$s_3^1 = 6, s_4^1 = 4$
t_8	$\{4,6\}$	$\{6\}$	$s_3^2 = 4, s_4^2 = 6$	t_9	$\{4\}$	$\{4\}$	$s_5^1 = s_4^1$
t_{10}	$\{6\}$	$\{2,6\}$	$s_4^2 = 6, s_5^2 = 2$	t_{11}	$\{0,4\}$	$\{0\}$	$s_5^1 = 4, s_6^1 = 0$
t_{12}	$\{2\}$	$\emptyset$	$s_5^2 = 2$				

Table I

DECODING OF THE TWO SIGNALS OF FIGURE 3.

or more precisely, during $[t_1; t_1 + \delta]$, the receiver is not able to detect the superposition of preambles anymore (due to the presence of up-chirps superposed with down-chirps). Thus, it knows that the preamble of n_1 has reached its first down-chirp at t_1.

Data decoding: We define the sequence of decoded data for n_1 by s^1 and the sequence of decoded data for n_2 by s^2. t_2, which is the beginning of the payload of n_2, is the first time where only up-chirps of data symbols are superposed. At frontier t_2, the receiver stores the current frequencies, which correspond to $F_+(t_2) = \{4, 6\}$. At frontier t_3, the receiver computes $F_-(t_3)$ by updating the previous frequencies $F_+(t_2) = \{4, 6\}$, and obtains $F_-(t_3) = \{2, 4\}$ (each frequency of $F_+(t_2)$ is increased by $3/4 \cdot 2^{SF} = 6$ since $3/4$ time units have passed since t_2). The receiver detects the current frequencies $F_+(t_3) = \{2, 4\}$. There is no change in the frequencies ($F_-(t_3) = F_+(t_3)$), since the beginning of the data of n_1 starts with

the repeated symbol 2. Thus, the algorithm leaves $*$ for the first symbol of n_1 (to be decoded later), so $s^1 = (*)$. At frontier t_4, the receiver computes $F_-(t_4)$ by updating the previous frequencies $F_+(t_3) = \{2, 4\}$, and obtains $F_-(t_4) = \{4, 6\}$ (since $1/4$ time units have passed). It detects the current frequencies $F_+(t_4)$, and obtains $F_+(t_4) = \{0, 4\}$. Thus, one frequency changed from 6 to 0, hence, $s^2 = (6, 0)$, since t_4 is a frontier of n_2. The current symbol of n_1 corresponds to frequency 4 (which is translated into 2 at the beginning of the symbol frontier of n_1, which was t_3). At frontier t_5, the receiver computes $F_-(t_5)$ by updating the previous frequencies $F_+(t_4) = \{0, 4\}$, and obtains $F_-(t_5) = \{2, 6\}$. It detects the current frequencies $F_+(t_5) = \{6\}$, which can also be written $\{6, 6\}$. The frequency of n_1 changed from 2 to 6, hence $s^1 = (*, 2, 6)$. The current symbol of n_2 corresponds to frequency 6 (which translates to 0 at the beginning of the symbol frontier of n_2, t_4, and was already known). The algorithm continues until t_{12}, where no frequency is received. Thus, the algorithm knows that all nodes have stopped their transmissions. The algorithm removes the last predicted symbol of n_1 (indeed, at t_{11}, it considered that n_1 was transmitting a symbol with the same frequency as the frequency of n_2). At this step, the decoded frames are $s^1 = (*, 2, 6, 4, 4)$ for n_1 and $s^2 = (6, 0, 4, 6, 2)$ for n_2. Then, the algorithm replaces all special values $*$ with the first known value of the frame by backtracking (since we know $s_2^1 = s_1^1$). The algorithm uses the frame length present in each frame to truncate the frames to their correct length. Finally, the algorithm outputs are $(2, 2, 6, 4, 4)$ and $(6, 0, 4, 6, 2)$, as expected.

Generalization of preamble detection and data decoding

In this paragraph, we generalize the example given above and we formulate our proposition in Algorithm 1.

Preamble detection: The superposition of the beginning of the preambles results in the superposition of up-chirp symbols. This superposition enables the receiver to detect two sharp frequency edges, each sharp edge allowing the receiver to know the symbol frontier of a transmitter. The beginning of the first data symbol of the first node is not decodable, as it corresponds to an up-chirp (for node n_1) superposed with a down-chirp (for the end of the preamble of n_2).

Data decoding: From the beginning of the first data symbol of the second node, only up-chirps are superposed, and thus it is possible to detect all sharp edges. The difficulty relies in correlating each frequency with the symbols of each node. To do so, we use the following property: sharp edges can occur only at the beginning of a symbol, when the symbol changes, or once during a symbol. When the sharp edge occurs during a symbol, it can be predicted if the symbol value is known.

Algorithm 1 describes our proposed algorithm. It starts after the superposed preambles have been received, and thus considers that the symbol frontier of each transmitter is known. The algorithm considers the frontiers of all data symbols sequentially, apart from the first frontier of the first node for which the frequency cannot be obtained. At each frontier, the receiver updates the previous frequencies (since frequencies change over time in LoRa chirps, and time has passed since the detection of the previous frequencies). Then, the receiver compares these (updated) previous frequencies F_- with the current frequencies F_+. Note that in practice, it may take up to δ time units to obtain the current frequencies, so the receiver might have to update the current frequencies based on the detection duration. Only two cases can occur for the algorithm.

Case 1: Exactly one frequency has changed. This can only happen when a new symbol starts, which can only occur at the symbol frontier. Since the receiver knows if the current frontier is for the first or the second transmitter, it knows the new symbol for the current node (based on the new frequency), the previous symbol for the current node (based on the frequency that has changed), and the current symbol for the other node (based on the frequency that did not change).

Case 2: No frequency has changed. This can only happen when the new symbol is equal to the previous symbol (this was the case on Figure 3 at times t_3 and t_9).

- If the receiver knows the previous symbol of the current node (time t_9 of Figure 3), the new symbol can be deduced.
- Otherwise, the previous symbol of the current node is unknown, which corresponds to the beginning of the algorithm when the first symbol is repeated (time t_3 of Figure 3). In this case, the algorithm leaves a special value (denoted by $*$ here). As soon as one symbol changes, the receiver is able to identify the new and previous symbols of the end-device corresponding to that frontier, and hence to deduce the symbol of the other end-device. In addition, the algorithm can replace all the $*$ values of the frame of the current node with the value of the previous symbol. This is why we assumed at least one symbol change per frame.

The time complexity of our algorithm is linear with the number of symbols of the longest frame. Most of the symbols are decoded on the fly, δ time units after the beginning of the symbol, except for the symbols repeated initially (see the last loop of the algorithm). The space complexity of our algorithm is $\mathcal{O}(1)$, since the storage requirement is limited to the value of the first non-special symbol for each node. Thus, the algorithm is extremely efficient

Algorithm 1: Decoding of two slightly desynchronized superposed LoRa signals.

```
for each frontier t_i of a data chirp do
    compute currentSymbol and currentNode
    if currentSymbol=0 and currentNode=1 then
        skip (frequencies cannot be detected)
    else
        F_+(t_i) ←detect current frequencies
        if currentSymbol=0 and currentNode=2 then
            skip (F_-(t_i) cannot be computed)
        else
            compute F_-(t_i) by updating F_+(t_{i-1})
            newF ← F_+(t_i) − F_-(t_i)
            oldF ← F_-(t_i) − F_+(t_i)
            if newF = ∅ then
                the new symbol in symb[currentNode] is equal to the previous (or to ∗)
            else
                the previous symb. in symb[currentNode] is equal to the value of oldF
                the new symbol in symb[currentNode] is equal to the value of newF
for each node n do
    replace in symb[n] all the leading ∗ values with the first defined value
    truncate the frame according to its length
```

in time and space, for two nodes.

C. Case of several slightly desynchronized signals

Note that with our assumptions, decoding three or more signals is not always possible. For instance, Figure 4 shows two sets of different signals that produce the same superposition of frequencies, and thus cannot be decoded.

Algorithm 2 describes our proposed algorithm, for three or more nodes. It is similar to Algorithm 1, with the following main changes. (1) When $F_-(t) = F_+(t)$ at the frontier of a node n, it is not possible to assume that the symbol of n remains the same. Indeed, if the number of frequencies of $F_-(t)$ is smaller than the number of nodes, the frequency of node n might have changed from one superposed frequency to another superposed frequency.

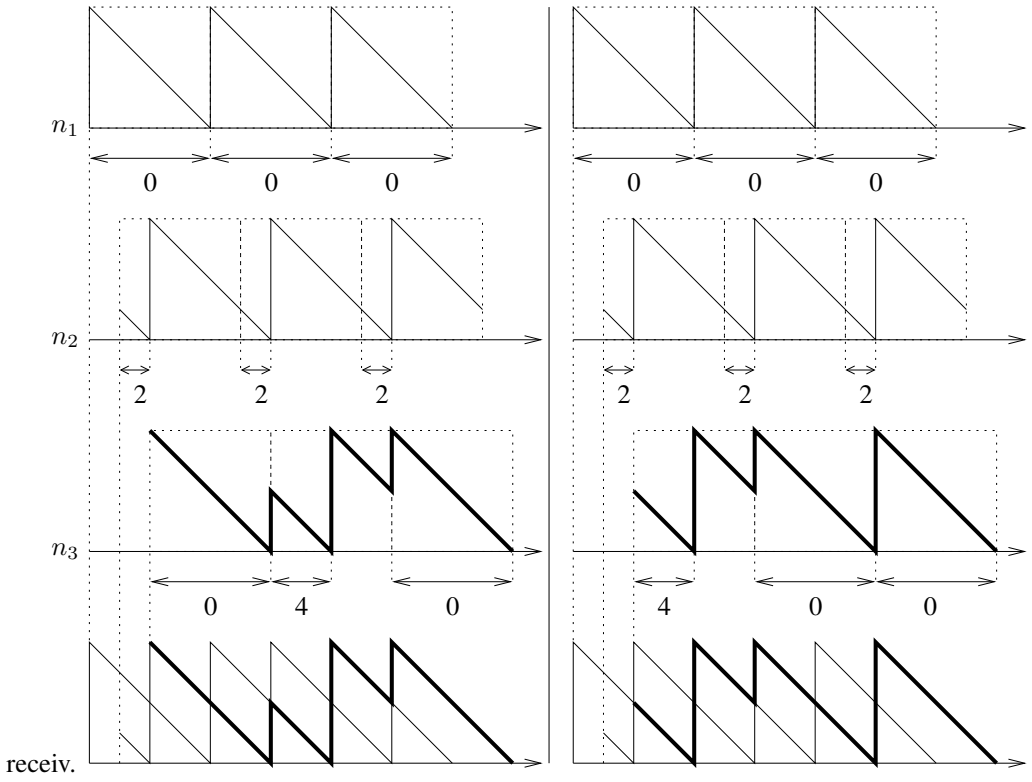

Figure 4. When three nodes that are slightly desynchronized transmit frames, it is not always possible to decode them: these two sets of frames produce the same superposition of frequencies.

(2) Consequently, initial repeated symbols which yielded unchanging frequencies cannot be decoded.

Algorithm 2 is able to decode many cases of slightly desynchronized signals for n transmitters, when $n \geq 3$, while Algorithm 1 is able to decode all cases of slightly desynchronized signals for $n = 2$ transmitters. It only fails to do so when the number of received frequencies is within $[2; n-1]$ (which never occurs when $n = 2$). Indeed, in this case, even if the algorithm knows that the frequency of the current node has changed, it cannot determine what is the new value, as it has $n - 1 > 1$ possibilities. It can still deduce the value of the previous symbol for this node. At the next frontier for this node, though, the value of this symbol might be deduced, depending on the number of other frequencies.

Figure 5 shows the superposition of three signals, and Table II shows the decoding of the three superposed signals of Figure 5, according to Algorithm 2. Initially, $F_+(t_2) = \{3, 4, 7\}$. Then, the algorithm computes $F_-(t_3) = \{0, 3, 7\}$ and obtains $F_+(t_3) = \{0, 4, 7\}$. The first symbol s_1^1 of node n_1 is thus 3, and the second symbol s_2^1 of node n_1 is 4. Then, the algorithm computes $F_-(t_4) = \{1, 2, 6\}$ and obtains $F_+(t_4) = \{1, 6\}$. The first symbol s_1^2 of node n_2 is 2, but it is not possible to determine the second symbol of node n_2 yet. Then, the algorithm

Algorithm 2: Decoding of three or more slightly desynchronized superposed LoRa signals.

for *each frontier* t_i *of a data chirp* **do**
  compute currentSymbol and currentNode
  if *currentSymbol=0 and currentNode*$\neq$ *lastNode* **then**
    skip (frequencies cannot be detected)
  else
    $F_+(t_i) \leftarrow$detect current frequencies
    if *currentSymbol=0 and currentNode=lastNode* **then**
      skip ($F_-(t_i)$ cannot be computed)
    else
      compute $F_-(t_i)$ by updating $F_+(t_{i-1})$
      $newF \leftarrow F_+(t_i) - F_-(t_i)$
      $oldF \leftarrow F_-(t_i) - F_+(t_i)$
      if $oldF \neq \emptyset$ **then**
        the previous symb. in $symb$[currentN.] is equal to the value of $oldF$
      if $newF \neq \emptyset$ **then**
        the new symbol in $symb$[currentNode] is equal to the value of $newF$

computes $F_-(t_5) = \{0, 3\}$ and obtains $F_+(t_5) = \{0, 3, 4\}$. The second symbol s_2^3 of node n_3 is 4, but it is not possible to determine whether the first symbol of node n_3 is 0 or 3. The algorithm continues until t_{17}.

time	F_-	F_+	symbol	time	F_-	F_+	symbol
t_2	unknown	$\{3,4,7\}$	initialization	t_3	$\{0,3,7\}$	$\{0,4,7\}$	$s_1^1=3, s_2^1=4$
t_4	$\{1,2,6\}$	$\{1,6\}$	$s_1^2=2$	t_5	$\{0,3\}$	$\{0,3,4\}$	$s_2^3=4$
t_6	$\{0,4,7\}$	$\{0,1,7\}$	$s_2^1=4, s_3^1=1$	t_7	$\{1,2,3\}$	$\{2,3,7\}$	$s_2^2=1, s_3^2=7$
t_8	$\{1,4,5\}$	$\{1,2,5\}$	$s_2^3=4, s_3^3=2$	t_9	$\{1,5,6\}$	$\{5,6\}$	$s_4^1=1$
t_{10}	$\{0,7\}$	$\{0,2\}$	$s_3^2=7, s_4^2=2$	t_{11}	$\{2,4\}$	$\{2,4\}$	?
t_{12}	$\{0,6\}$	$\{0,6\}$	?	t_{13}	$\{0,2\}$	$\{0,2\}$	?
t_{14}	$\{2,4\}$	$\{0,2\}$	$s_4^3=4, s_5^3=0$	t_{15}	$\{4,6\}$	$\{4,6\}$	?
t_{16}	$\{0,6\}$	$\{6\}$	$s_5^2=0, s_6^2=6$	t_{17}	$\{0\}$	$\emptyset$	$s_5^3=0$

Table II
PARTIAL DECODING OF THE THREE SIGNALS OF FIGURE 5.

Table III shows the output of Algorithm 2. The frame of n_2 is successfully decoded. However, the frame of n_1 has its last two symbols unknown, and the frame of n_3 has its first

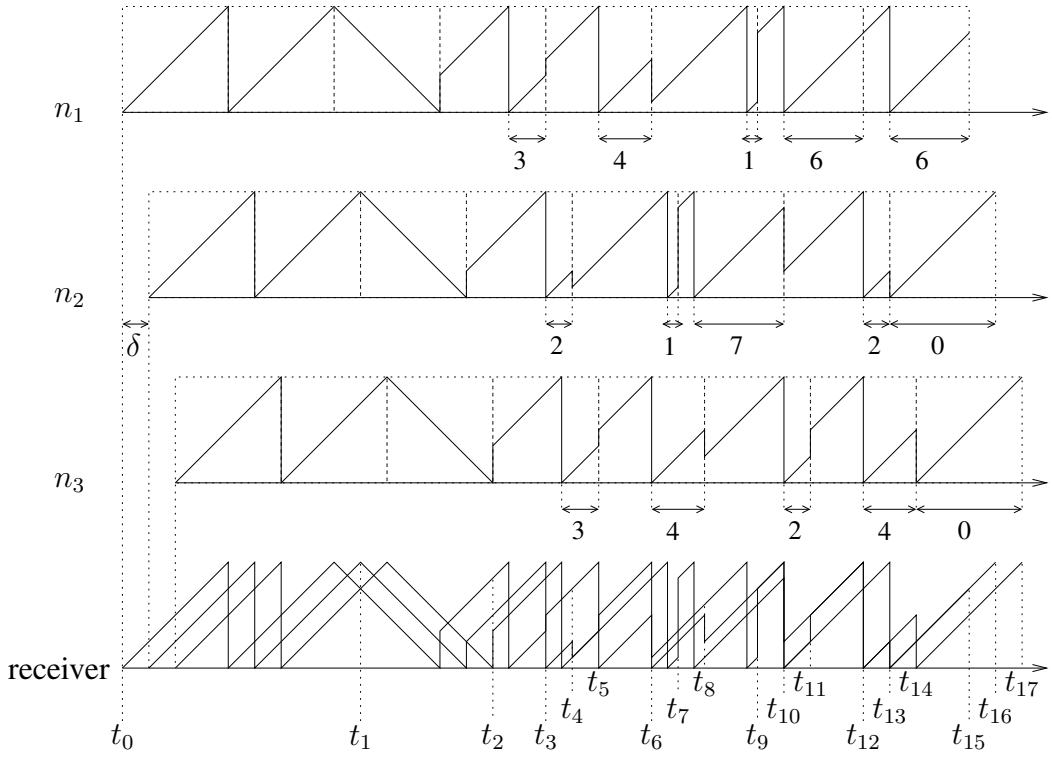

Figure 5. The superposition of three signals produce a very complex signal, which can be partially decoded.

symbol unknown.

node	symbol 1	symbol 2	symbol 3	symbol 4	symbol 5
n_1	3	4	1	$\{5,6\}$	$\{0,6\}$
n_2	2	1	7	2	0
n_3	$\{0,3\}$	4	2	4	0

Table III
OUTPUT OF ALGORITHM 2 ON THE SIGNALS OF FIGURE 5. ONLY ONE FRAME IS COMPLETELY DECODED.

D. Cyclic Redundancy Check for decoding

It is possible to use the CRC present in each frame in order to improve the decoding rate of Algorithm 2.

Let us consider the output of Table III as an example. The first symbol of the frame of n_3 is unknown, but the uncertainty is limited to two possible values for this symbol. Thus, the frame for n_3 is either (0,4,2,4,0) or (3,4,2,4,0). We can verify the CRC value for each possible frame: if only one frame has a correct CRC, then this frame is the correct frame. If both frames have a correct CRC, which is possible but unlikely, then the frame cannot be decoded. Similarly, the possible frames for n_1 are either (3,4,1,5,0), (3,4,1,6,0), (3,4,1,5,6) or

(3,4,1,6,6). Since there are more uncertainties, the probability of having at least two frames with a correct CRC is higher, and it is less likely that this frame can be decoded. In order to avoid having to compute a large number of CRCs (with limited decoding performances), we set a limit to how many CRCs are performed per frame.

In order to show the performance of using the CRC in our MAC protocol, we consider the following scenario. We force situations where Algorithm 2 occurs by ensuring that all end-devices send a colliding frame with a slight desynchronization. We set the frame size to 50 bytes, the SF to 7 and we set the number of CRC attempts per frame to 4 or 100. We implemented random symbols for the frames, and the actual CRC algorithm of the LoRaWAN standard, which is CCITT-16 (see Subsection 15.2 of [10]).

Figure 6 shows the average number of CRC attempts per frame. We notice that the number of needed CRC increases with the number of collided frames. This is because the more colliding signals, the more uncertainties there are in frames. Moreover, we notice that with a threshold of 4, the CRC algorithm cannot be applied for more than $n = 5$ colliding frames, due to a large number of uncertainties. In this case, each frame needs more than 4 CRC to be decoded, so no CRC is actually computed. However, frames are able to be decoded by increasing the number of authorized CRCs per frame, e.g. to 100.

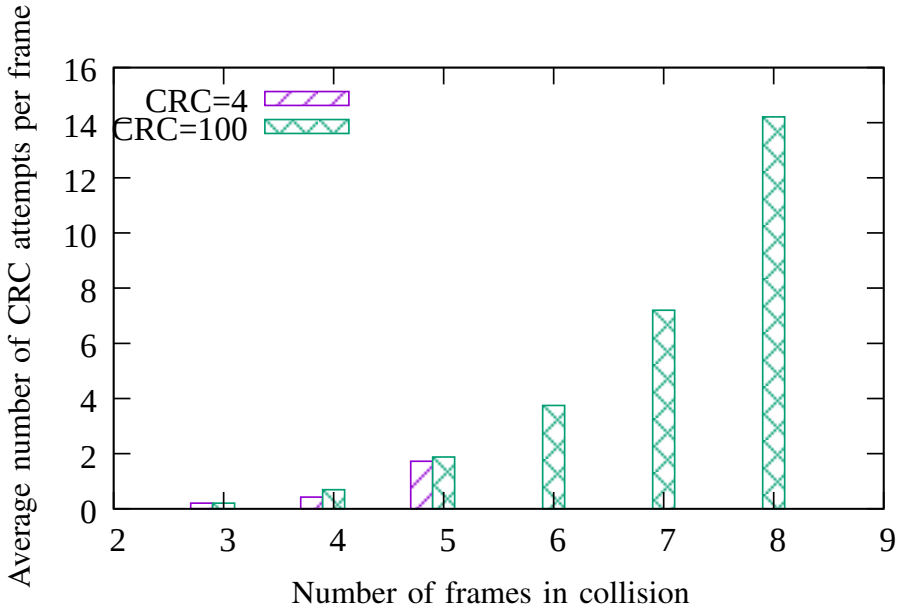

Figure 6. The number of CRCs needed to decode a frame increases with the number of frames in collisions.

Figure 7 shows the number of decoded frames with and without CRC for the scenario described above. We can notice that for a small number of authorized CRCs per frame such as 4, we see a small improvement when the number of colliding frames is less than or equal to 5. Above this number, the CRC algorithm is not able to decode frames and thus, it has

the same behaviour as if CRC were disabled (case of CRC=0). However, by increasing the number of authorized CRCs per frame, we can notice that the number of decoded frames increases slightly and can improve the throughput up to 8% when eight frames are colliding.

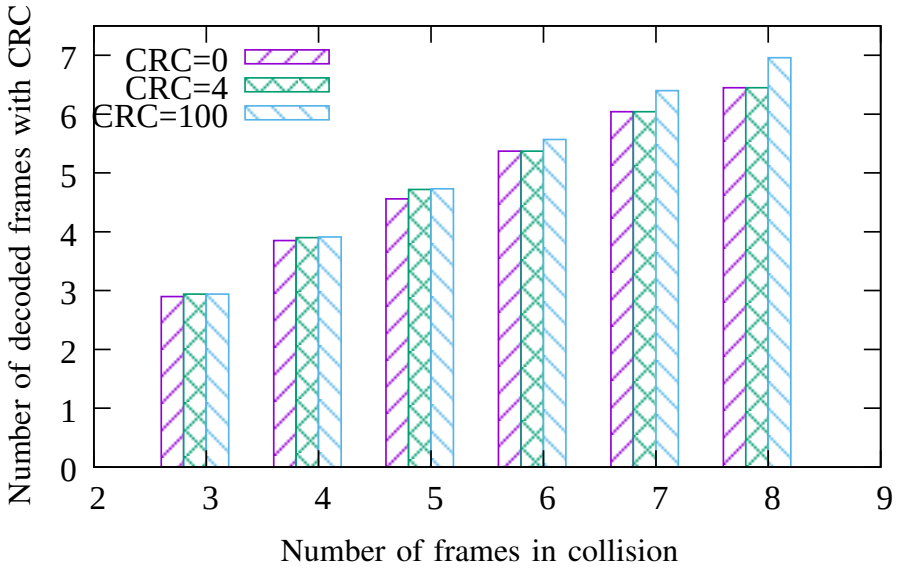

Figure 7. The CRC algorithm increases the number of decoded frames.

IV. Proposed Collision Resolving MAC protocol

In this section, we present a new MAC protocol which enables slightly desynchronized LoRa signals. Then, we provide an analysis of this proposed MAC protocol.

A. Protocol Description

Algorithm 1 and Algorithm 2 require transmissions to be slightly desynchronized, by less than one symbol, which is a rare event in LoRaWAN. Thus, we designed a new MAC protocol called Collision Resolving-MAC (CR-MAC).

The CR-MAC protocol works as follows. Each gateway sends periodic beacons on each SF. These beacons are sent simultaneously by all gateways, as in Class B of LoRaWAN. Upon receiving a beacon, each end-device starts S consecutive slots, whose duration is equal to the maximum frame transmission plus one symbol. To transmit a frame, an end-device has to wait for the beginning of a slot. It then draws a random number between 0 and $s = (SD/\delta) - 1$, and delays its transmission by $s \times \delta$. We call sub-slots the possible starting times within each slot.

Figure 8 depicts the CR-MAC protocol. There are three beacons, and $S = 3$ slots after each beacon. At the beginning of each slot, there are $s = 4$ sub-slots, which correspond to possible starting times for the transmission of frames.

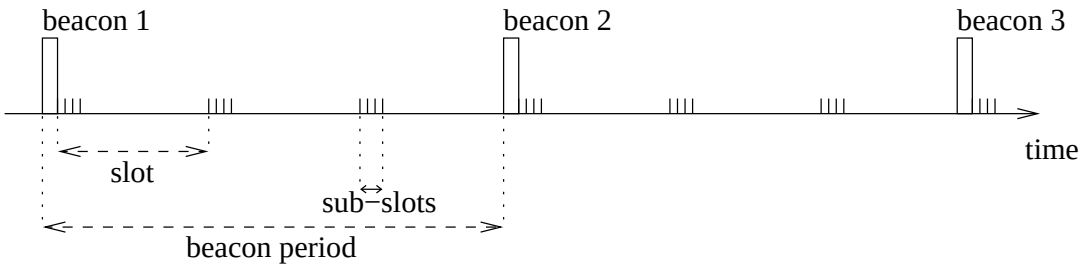

Figure 8. Our proposed CR-MAC protocol.

With the CR-MAC protocol, if n end-devices decide to transmit a frame on the same channel, with the same SF and on the same slot, the probability that these transmissions are slightly desynchronized is equal to the probability that each node chooses a different sub-slot. This probability increases with s, and decreases with n.

If several end-devices transmit during the same sub-slot, Algorithm 2 fails. However, this can be detected by counting the number of frequencies in the set $newF$ in Algorithm 1: if it is equal to two or more, there are multiple transmissions in the same sub-slot.

In practice, the number of slots S depends on the clock drift of the end-devices, and on the symbol duration. S has an impact on the energy efficiency of CR-MAC, as it requires end-devices to listen to the beacon. Note that it would be possible for the node to listen to the beacon only when it has a frame to transmit, but in this case, S would have a larger impact on the delay.

The number of sub-slots s is computed as the symbol duration divided by δ. The value of s gives an upper bound on the number of frames that can be decoded by Algorithm 2, or equivalently on the number of slightly desynchronized transmissions. Recall that δ is fixed by the hardware capabilities.

The impact of this protocol on the energy consumption is limited to end-devices listening to periodic beacons, and to a slight increase in the delay before a transmission (this delay is smaller than the slot duration).

B. Analysis of the Proposed CR-MAC protocol

The performance of the CR-MAC protocol is closely related to the number of collisions, namely the number of end-devices choosing the same slot but different sub-slots.

If we denote by n the number of end-devices that choose the same slot and by s the number of sub-slots in the same slot, the probability that all n end-devices choose a distinct sub-slot $p(n, s)$ can be determined as follows:

- if $n > s$: at least two end-devices will choose the same sub-slot, therefore $p(n, s) = 0$,
- if $n \leq s$: the number of possible patterns where all n end-devices choose distinct sub-slots is equal to the number of arrangements of n among s, defined by the number of combinations of n elements among s with ordering of n, i.e. $A_s^n = C_s^n \times n!$, and the total number of patterns is equal to s^n, giving:

$$p(n, s) = \frac{A_s^n}{s^n} = \frac{s!}{(s-n)!s^n}. \quad (1)$$

This probability depends on s, which is limited by the hardware, and on n, which depends on the total number of end-devices and on their duty-cycle. Thus, if s is small, it is important to reduce n for CR-MAC to achieve a good performance.

Table IV shows the probability that all end-devices have different sub-slots, for several values of n and of s. Obviously, the probability is 0 when there are more end-devices than sub-slots. As the number of sub-slots increases, the probability increases. For instance, the probability that $n = 4$ end-devices have different sub-slots is about 9% for $s = 4$, and is 41% for $s = 8$. As the number of end-devices increases, however, the probability decreases. For instance, for $s = 8$, the probability decreases from 41% for $n = 4$ end-devices to about 2% for $n = 7$ end-devices. Thus, it is very important that the number of end-devices sharing the same slot is kept low, ideally between 2 and $n = s/2$. In practice, this can be controlled by reducing the duty-cycle of end-devices.

Number of end-devices n	Probability for $s = 2$	Probability for $s = 4$	Probability for $s = 8$
2	0.5	0.75	0.875
3	0	0.375	0.656
4	0	0.094	0.410
5	0	0	0.205
6	0	0	0.077
7	0	0	0.019
8	0	0	0.002

Table IV

NUMERICAL APPLICATION FOR THE PROBABILITY THAT THE n END-DEVICES HAVING CHOSEN THE SAME SLOT ALSO CHOOSE DIFFERENT SUB-SLOTS, AS A FUNCTION OF n AND OF THE NUMBER OF SUB-SLOTS s.

V. NUMERICAL RESULTS

In this section, we evaluate and compare the network performance in terms of system throughput, energy efficiency, and system delay for both the conventional LoRaWAN protocol and our CR-MAC protocol.

A. Parameter settings

Simulations are carried out using our own simulator developed in Perl. We considered only one network server and one gateway in the network. We set the number of allowed CRC computation per frame to 4 and the size of preamble for each frame to 6 symbols (which becomes 10.25 after the addition of 2 symbols for the sync word and 2.25 symbols for down-chirps). For some simulations, we vary the number of end-devices but we set the size of the sent frames to 50 bytes. For other simulations, we vary the size of sent frames but we set the number of end-devices to 100. We also consider that all end-devices have a duty cycle of 1% and are on the same channel with the same SF without capture conditions[1]. We set the bandwidth to 125 kHz in order to have a fair comparison of the delay as it depends on the bandwidth and on the SF [1]. We also set the number of slots in a beacon period to 100 for our CR-MAC protocol unless otherwise specified and we consider a beacon size of 10 bytes. Finally, in case of collision, we consider only one retransmission for successful reception, which is an ideal condition for conventional LoRaWAN[2].

B. Throughput

Figure 9 shows the percentage of successfully decoded frames as a function of the size of the sent frames for both the conventional LoRaWAN and our MAC protocol called CR-MAC. We vary the number of sub-slots to 2, 4, and 8. In the conventional LoRaWAN protocol, when several transmissions overlap, the signals collide and are thus considered lost. However, in the CR-MAC protocol, when more than one end-device use the same slot, their signals might be decoded if each end-device uses a different sub-slot. Thus, increasing the number of sub-slots reduces destructive collisions and increases the throughput of the system. It can be seen that the CR-MAC protocol outperforms the conventional LoRaWAN protocol. Indeed, the throughput computed by LoRaWAN is about 40%, while it reaches 58% using CR-MAC with two sub-slots, 76% with four sub-slots, and 83% with eight sub-slots.

Figure 10 shows the percentage of successfully decoded frames in terms of the number of end-devices in the network for both LoRaWAN and CR-MAC protocols. We notice that this percentage decreases by increasing the number of end-devices for both protocols. This is due to the fact that in large networks, collisions are more important than in small networks.

[1]Note that, under capture conditions the performance of the network will be improved for both LoRaWAN and CR-MAC protocols.

[2]Note that a successful decoding after the first retransmission is more likely to happen with CR-MAC than with LoRaWAN, as shown later in Subsection V-B.

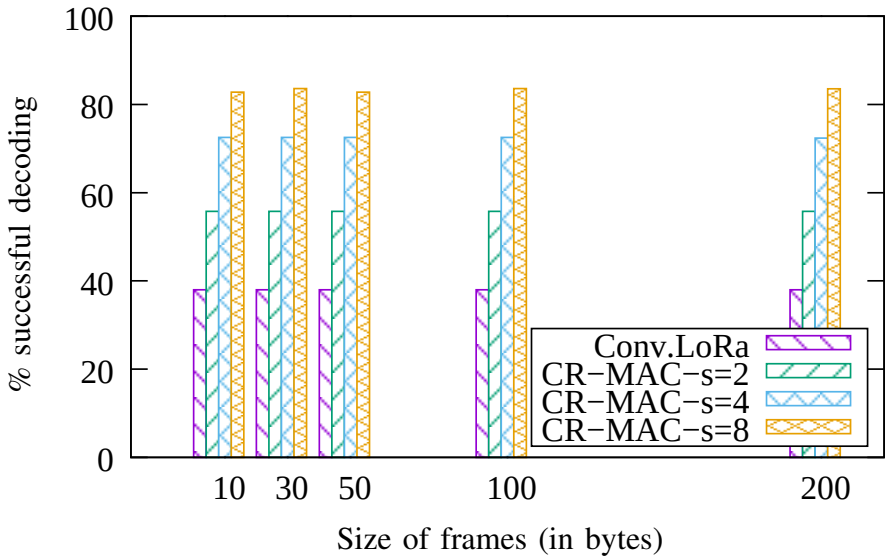

Figure 9. The percentage of successfully decoded frames increases by increasing the number of sub-slots.

We can also notice that the performance of LoRaWAN degrades consistently compared to CR-MAC for both spreading factors SF7 and SF12. The percentage of loss is about 9 times lower in large networks (250 end-devices) than in small networks (10 end-devices). This percentage of loss is less drastic using CR-MAC. Indeed, in small networks, it is almost 0% and even for large networks, the throughput of the system goes up to 52% with SF7 and 55% for SF12. This is due to the fact that, with our proposed superposed signal decoding technique, CR-MAC is able to resolve many colliding frames.

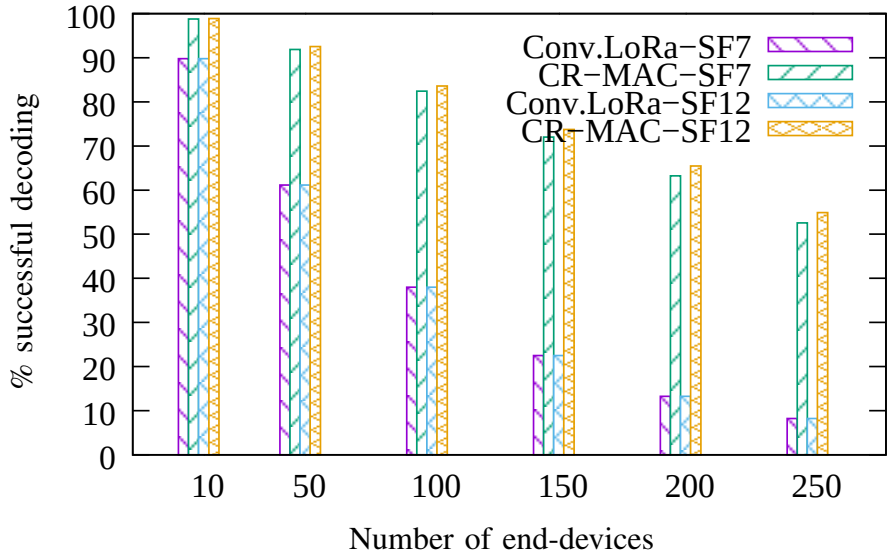

Figure 10. The collision resolution technique yields to considerably increasing the percentage of decoded frames.

Nb. End-devices	Conv.LoRa SF7	CR-MAC SF7	Conv.LoRa SF12	CR-MAC SF12
10	5428.571×10^3	11000×10^3	28.16×10^3	173.945×10^3
50	203.301×10^3	1107.754×10^3	1049	3290
100	40.761×10^3	267.908×10^3	209	1655
150	12.772×10^3	103.184×10^3	65	619
200	5.055×10^3	52.921×10^3	25	316
250	2.388×10^3	27.777×10^3	12	163

Table V

THE ENERGY EFFICIENCY (IN BPJ) IS REDUCED BY INCREASING THE NUMBER OF END-DEVICES IN THE NETWORK.

Figure 11 shows the average throughput in terms of the number of end-devices in the network for both LoRaWAN and CR-MAC protocols. We notice that the average throughput decreases with the number of end-devices, as in large networks, collisions occur more frequently. Moreover, we notice a large gain between the throughput computed with SF7 and the throughput computed with SF12. Indeed, although a larger SF enables slightly better collision resolution, it corresponds to much smaller bitrates. Finally, as our protocol is able to decode superposed LoRa signals, we notice that it outperforms LoRaWAN protocol with a gain up to 75%.

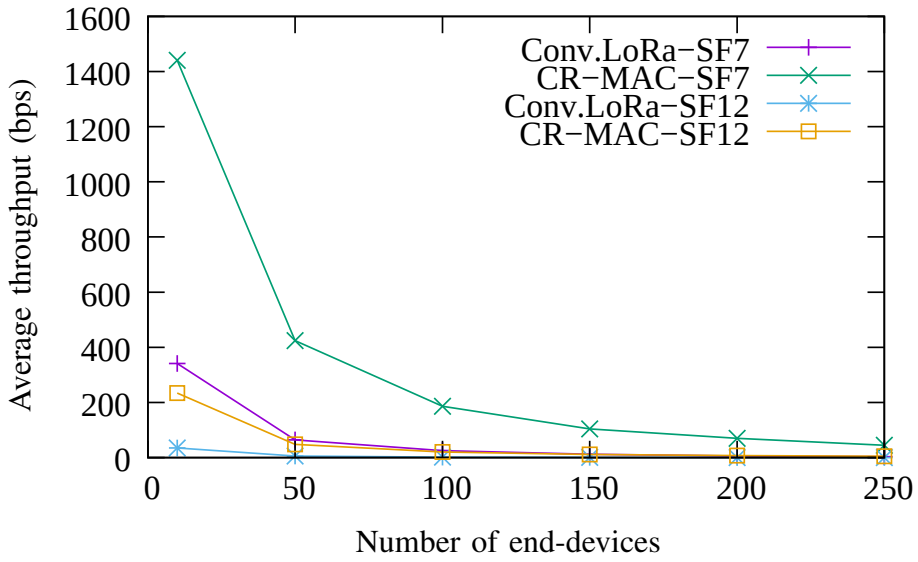

Figure 11. The collision resolution technique yields to considerably increasing the average throughput of the network.

C. Energy Efficiency

In this subsection, we compute the energy efficiency defined by the ratio of the total number of successfully received bits and the total consumed energy. The consumed energy

for LoRaWAN is the sum of transmit powers during frame transmission for all the end-devices. However, for CR-MAC, the consumed energy is the sum of the transmit powers during frame transmission for all end-devices and the power required for listening beacons during the time on air of beacons for a beacon size of 10 bytes.

Table V shows the energy efficiency computed for both LoRaWAN and CR-MAC protocols, for SF7 and SF12. We set the number of slots to 100 and we vary the number of end-devices in the network. The transmit power is set to 66 mW and the received power is set to 19.5 mW [24]. We notice that using SF7, CR-MAC protocol outperforms LoRaWAN with a gain up to 50% for small networks and up to 90% for large networks. Moreover, using SF12, the CR-MAC protocol shows a large gain compared to LoRaWAN. The gain goes up to 84% for small networks and up to 92% for large networks. This is due to the fact that the collision resolution implemented by CR-MAC reduces the delay as the number of retransmissions decreases compared to LoRaWAN, and considerably increases the average throughput.

Figure 12 and Figure 13 show the energy efficiency computed for both LoRaWAN and CR-MAC protocols. Here, we set the number of end-devices to 100 and we vary the number of slots in the beacon period. The energy efficiency computed by LoRaWAN is always the same. This is because transmissions in LoRaWAN follow the ALOHA mechanism and are independent of slots. However, we notice that the energy efficiency computed by CR-MAC slightly increases with the number of slots especially with SF7. Indeed, frames transmitted with SF7 have a short time on air. Thus, end-devices are in listening mode frequently and the CR-MAC protocol generates more beacon periods. This is why the energy efficiency is less important with a small number of slots than the energy efficiency achieved with a large number of slots. However, frames transmitted with SF12 have a large time on air and thus CR-MAC protocol results into less beacon periods per unit time, compared to SF7. This is why the energy efficiency remains almost constant against the number of slots in a beacon period. Compared to LoRaWAN, we observe a gain between 72% for a small number of slots and 86% for a large number of slots using SF7, and a gain of 87% using SF12.

D. Delay

Figure 14 shows the average delay in terms of the number of end-devices for LoRaWAN and CR-MAC protocols. We notice that the delay increases with the number of end-devices and SF for both protocols. Indeed, in conventional LoRaWAN with a large number of end-devices, the probability to send frames without interference is low as end-devices use ALOHA mechanism for transmission. We observe also that CR-MAC outperforms LoRaWAN protocol

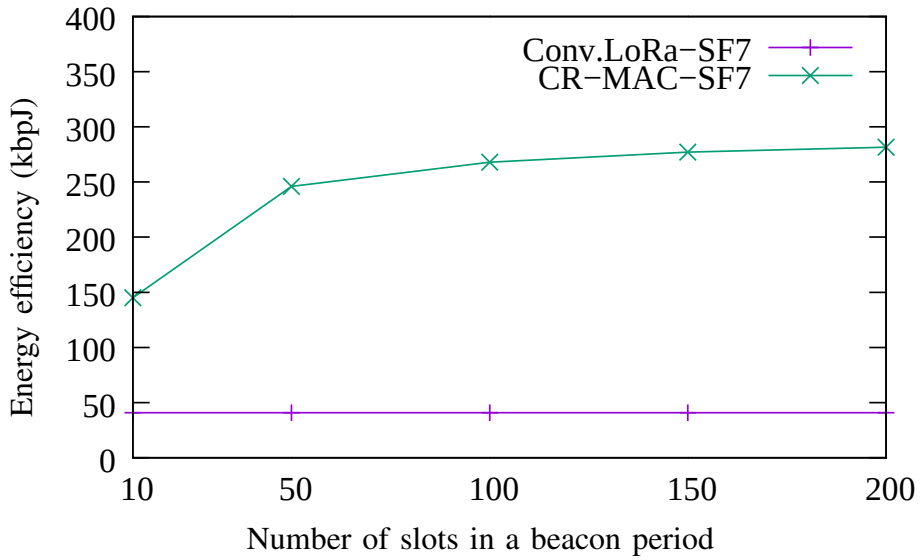

Figure 12. Using SF7, the transmission time is small, thus the end-devices listen frequently. This reduces the energy efficiency. Despite this, CR-MAC outperforms LoRaWAN.

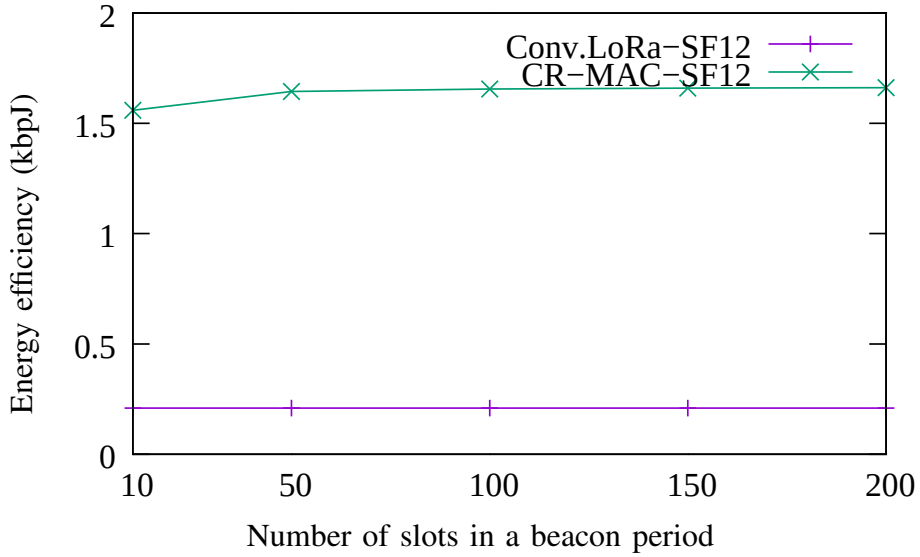

Figure 13. Using SF12, the transmission time is large, thus the listening period of the end-devices is short. This does not significantly impact the energy efficiency. CR-MAC outperforms LoRaWAN.

and shows a delay reduction of 10% for small networks and of up to 45% for large networks. This is due to the fact that CR-MAC reduces the percentage of frame collisions as it is beacon-based. Moreover, CR-MAC is able to correctly decode collided frames which is not the case of LoRaWAN protocol, and thus CR-MAC may reduce the number of retransmissions. Furthermore, we notice a difference in delay when using different spreading factors. Indeed, as our protocol is able to cancel collisions, retransmissions are not always needed. In addition,

the frame transmission duration depends on SF. With SF12, the transmission duration of a frame is larger than with SF7, thus inducing a larger delay for correct frames reception compared to SF7.

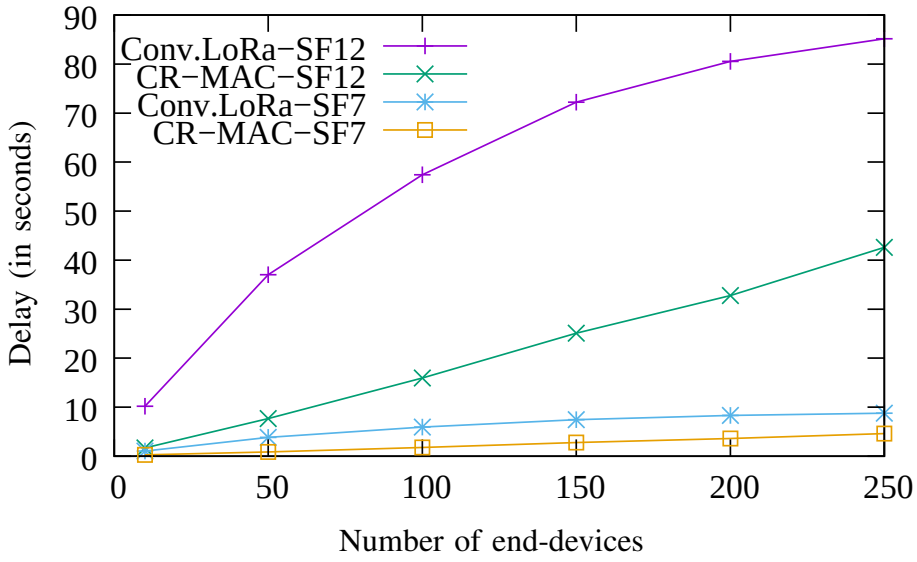

Figure 14. The delay against the number of end-devices for CR-MAC outperforms the delay for LoRaWAN due to our collision resolution technique.

Finally, Figure 15 shows the average delay in terms of the size of the frames for LoRaWAN and CR-MAC protocols. We notice that the delay increases with SF and with the size of the frame for both protocols. Indeed, dealing with large frames yields to long transmissions and thus long duration for channel unavailability for each end-device. For example, for SF7, a frame of 10 bytes needs 39.17 ms to be transmitted, while a frame of 100 bytes needs 172.29 ms. Moreover, a frame of 50 bytes needs about 2 seconds to be transmitted with SF12, but it needs only 95 ms with SF7. The time on air of frames is the same for LoRaWAN and for CR-MAC, but CR-MAC requires an extra delay of half a slot (to wait for the slot start) plus half a symbol (to wait for the sub-slot start) on average. However, CR-MAC still outperforms LoRaWAN in Fig 15 even under ideal retransmission conditions. In reality, the gain may be even higher because LoRaWAN might use the 7 retransmissions defined in [22] which is not the case for CR-MAC as it is able to decode superposed signals using the collision resolution technique. We notice a reduction of the delay of 75% for large frames.

To summarize, by resolving collisions, the CR-MAC protocol is able to jointly increase the network throughput and decrease the delay with a very small energy increment due to the beacons. Thus, the energy efficiency of CR-MAC is much higher than that of conventional LoRaWAN. In addition, smaller SFs further improve the achievable network performance.

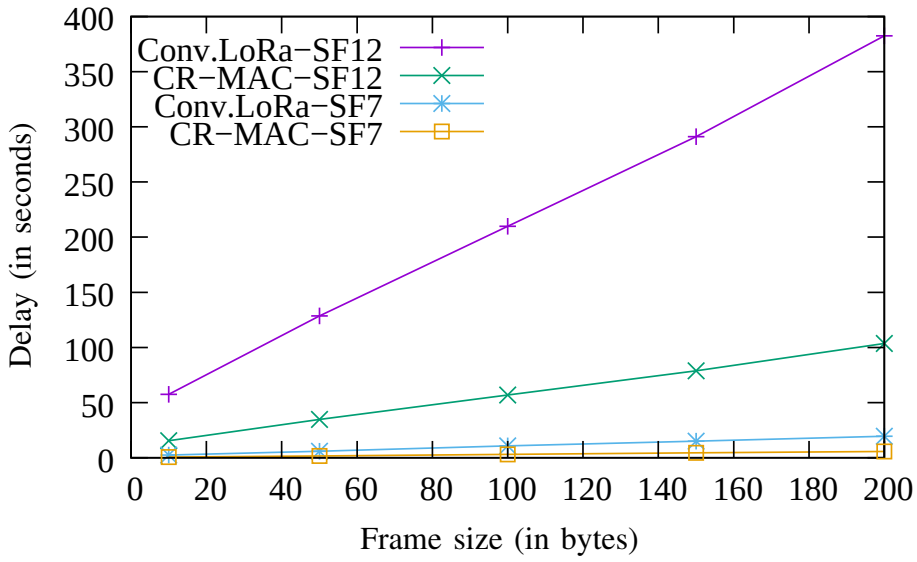

Figure 15. The delay against frame size for CR-MAC outperforms the delay for LoRaWAN due to our collision resolution technique.

VI. Conclusions

Collisions in LoRa networks are very harmful to the overall network performance. Indeed, when a gateway receives several superposed LoRa signals with comparable receive power levels, on the same channel and with the same SF, it is unable to decode these signals which are hence lost. In this paper, we proposed a novel beacon-based MAC protocol using a collision resolution technique that enables to decode two or more superposed LoRa signals. The proposed decoding algorithm exploits the slight desynchronization among superposed signals as well as the specificities of LoRa physical layer. We also show that the decoding performance of our collision resolution technique can be further improved by making use of the CRC which is already available in each frame. Simulation results show that, compared to the conventional LoRaWAN protocol, the proposed CR-MAC protocol provides remarkable performance improvements, both in terms of system throughput and energy efficiency. In addition, the proposed protocol enables significant delay reductions which is one of the most challenging tasks in 5G wireless communication systems.

In the future work, we will further enhance our proposed protocol by designing tailored retransmission strategies. Furthermore, the feasibility of our proposal may be demonstrated through experimental evaluations using Software Defined Radio.

REFERENCES

[1] Semtech Corporation, "AN1200.22 LoRa Modulation Basics," Semtech, Application note Revision 2, 2015, accessed 2018-01-29. [Online]. Available: http://www.semtech.com/uploads/documents/an1200.22.pdf

[2] Sigfox, http//www.sigfox.com.

[3] Ingenu, http//www.ingenu.com.

[4] X. Xiong, K. Zheng, R. Xu, W. Xiang and P. Chatzimisios, "Low Power Wide Area Machine-to-Machine Networks: Key Techniques and Prototype," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 64–71, September 2015.

[5] U. Raza, P. Kulkarni, R. Xu and M. Sooriyabandara, "Low Power Wide Area Networks: an Overview," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 2, pp. 855–873, January 2017.

[6] S. Kartakis, B.D. Choudhary, A.D. Gluhak, L. Lambrinos and J.A. McCann, "Demystifying low-power wide-area communications for city IoT applications," in *ACM Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization (WiNTECH '16)*, NYC, New York, October 2016.

[7] M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi, "Long-range communications in unlicensed bands: the rising stars in the IoT and smart city scenarios," *IEEE Wireless Communications*, 2016.

[8] K. E. Nolan, W. Guibene, and M. Y. Kelly, "An evaluation of low power wide area network technologies for the Internet of Things," in *International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 439–444.

[9] J. Petäjäjärvi, K. Mikhaylov, R. Yasmin, M. Hämäläinen, and J. Iinatti, "Evaluation of LoRa LPWAN technology for indoor remote health and wellbeing monitoring," *International Journal of Wireless Information Networks*, vol. 24, no. 2, pp. 153–165, 2017.

[10] LoRa Alliance Technical Committee, "LoRaWAN 1.1 Specification," LoRa Alliance, Standard V1.1, 2017.

[11] ——, "LoRaWAN 1.1 Regional Parameters," Standard V1.1, Revision A, 2017.

[12] Z. Qin and J. A. McCann, "Resource Efficiency in Low-Power Wide-Area Networks for IoT Applications," in *IEEE Global Communications Conference (Globecom)*, 2017.

[13] W. M. B. Reynders and S. Pollin, "Power and spreading factor control in low power wide area networks," in *IEEE International Conference on Communications (ICC)*, 2017.

[14] J. Lim and Y. Han, "Spreading Factor Allocation for Massive Connectivity in LoRa Systems," *IEEE Communications Letters*, vol. 22, no. 4, pp. 800–803, April 2018.

[15] M. Bor, U. Roedig, T. Voigt and J.M. Alonso, "Do LoRa Low-Power Wide-Area Networks Scale?" in *ACM MSWiM*, Malta, November 2016.

[16] O. Georgiou and U. Raza, "Low Power Wide Area Network Analysis: Can LoRa Scale?" *IEEE Wirel. Commun. Letters*, vol. 6, no. 2, pp. 162–165, April 2017.

[17] D. Croce, M. Gucciardo, I. Tinnirello, D. Garlisi, and S. Mangione, "Impact of spreading factor imperfect orthogonality in LoRa communications," in *International Tyrrhenian Workshop on Digital Communication (TIWDC)*, ser. Communications in Computer and Information Science (CCIS), vol. 766, 2017, pp. 165–179.

[18] G. Zhu, C.-H. Liao, M. Suzuki, Y. Narusue, and H. Morikawa, "Evaluation of LoRa receiver performance under co-technology interference," in *IEEE Consumer Communications and Networking Conference (CCNC)*, 2018.

[19] A. Waret, M. Kaneko, A. Guitton and N. El Rachkidy, "LoRa Throughput Analysis with Imperfect Spreading Factor Orthogonality," arXiv:1803.06534 [cs.NI], 2018.

[20] N. El Rachkidy, A. Guitton, and M. Kaneko, "Decoding superposed LoRa signals," in *IEEE Local Computer Networks (LCN)*, 2018.

[21] C. Goursaud and J.-M. Gorce, "Dedicated networks for IoT: PHY / MAC state of the art and challenges," *EAI endorsed Transactions on Internet of Things*, 2015.

[22] N. Sornin, M. Luis, T. Eirich, T. Kramp, and O. Hersent, “LoRaWAN Specification,” LoRa Alliance, Standard V1.0, 2015.

[23] J. Haxhibeqiri, F. Van den Abeele, I. Moerman, and J. Hoebeke, “LoRa scalability: a simulation model based on interference measurements,” *Sensors*, vol. 17, no. 6, p. 1193, 2017.

[24] Semtech Corporation, “SX1272/73 - 860 MHz to 1020 MHz Low Power Long Range Transceiver,” Semtech Corporation, Tech. Rep., March 2017, rev. 3.1.