Recovering Colliding LoRa Frames from Uncertainties Using LoRa Coding

Weixuan Xiao, Nancy El Rachkidy, Alexandre Guitton

Recovering Colliding LoRa Frames from Uncertainties Using LoRa Coding

Weixuan Xiao, Nancy El Rachkidy, Alexandre Guitton
Université Clermont Auvergne, CNRS, LIMOS, F-63000 Clermont-Ferrand, France
weixuan.xiao@uca.fr, nancy.el_rachkidy@uca.fr, alexandre.guitton@uca.fr

***Abstract*—LoRa is one of the leading technologies for Low-Power Wide Area Networks and the Internet of Things. Collisions in LoRa might cause retransmissions, which negatively impact the network performance and scalability. Several algorithms have been proposed to decode colliding frames under specific conditions. However, there remain indistinguishable frames due to uncertainties in some or all symbols. In this paper, we propose a general algorithm that significantly improves the recovery capabilities of existing algorithms by leveraging the LoRa coding techniques. Simulation results show that our algorithm can significantly reduce the number of the failed decoding of LoRa frames and improve the performance of the network.**

***Index Terms*—LoRa, LoRaWAN, LPWAN, LoRa coding, Collision resolution.**

I. Introduction

Long-range communication technologies such as LoRa [1], Sigfox [2], and Ingenu [3], can establish Low-Power Wide Area Networks (LPWANs). These technologies are becoming attractive choices for Internet of Things (IoT) applications because of the low energy consumption and large area coverage.

LoRa is a recent physical layer for LPWANs, which uses Chirp Spread Spectrum (CSS) modulation. CSS enables the simultaneous reception on different channels with different spreading factors (SFs). Based on LoRa, LoRaWAN is a simple MAC protocol with open specifications. It allows end devices to communicate to a network server via LoRa gateways. LoRaWAN has a limited throughput: its bitrate varies from 250 to 11000 bps due to energy-saving considerations. Constraints from using unlicensed radio bands also limit it. In Europe, for instance, end devices can only use a small duty-cycle (typically 1%) [4]. In a LoRa network, frames sent simultaneously by several end devices on the same channel, with the same SF might collide at the gateway. Such collisions further reduce the network performance as end devices might have to retransmit the frames in confirmed traffic.

Recently, many works have focused on the collision resolution of LoRa signals. Authors in [5] proposed a protocol to decode colliding frames from LoRa signals. They leverage the slight frequency offsets among transmitters in order to separate the symbols. The frequency offsets come from the oscillators of transmitters, due to the natural hardware imperfections. [6] [7] proposed to leverage the capture effect of LoRa to perform successive interference cancellation (SIC). When several signals are superposed, and one of them is captured, SIC algorithms reconstruct the strongest LoRa signal from the captured and decoded LoRa frame, remove this signal from the superposed signals, and process iteratively with the remaining signals. However, SIC algorithms cannot handle multiple signals with similar energy levels. Authors in [8] proposed the Generalized Slotted MAC (GS-MAC) protocol which enables LoRa gateways to decode frames that are slightly desynchronized, by a fraction of a symbol. Transmissions in GS-MAC start within one symbol of the beginning of a slot, in a random sub-slot. GS-MAC can decode most frames that are sent alone in their sub-slot, but generates uncertainties for the frames sent in the same sub-slot. Similar decoding algorithms were proposed in [9] [10].

This paper proposes an algorithm that can be applied to all the existing algorithms decoding colliding LoRa signals. Indeed, our algorithm takes as input the uncertainties produced by these existing algorithms, and reduces them using specific information from the LoRa coding (namely, the Hamming code, the interleaving, and the cyclic redundancy check). In many cases, our algorithm can completely remove the uncertainties and thus recover additional colliding frames. In this way, several retransmissions can be avoided in confirmed traffic, and the overall network performance is increased.

The structure of this paper is as follows. Section II describes LoRa, LoRa coding, and the LoRaWAN MAC protocol. Section III presents several collision resolution algorithms and the uncertainties from them. Section IV describes the system model and our proposed algorithm. Section V presents our simulation results. Section VI discusses the limitations of our algorithm. Finally, Section VII concludes the paper and provides prospects for future work.

II. LoRa and LoRaWAN

In this section, we first describe the LoRa physical layer. Second, we present the coding techniques in LoRa. Finally, we give an introduction to LoRaWAN, a widely used MAC protocol in LoRa networks.

A. LoRa

LoRa [11] is a physical layer technology for LPWAN based on a CSS modulation. LoRa frame is composed of a sequence of chirps. Each chirp consists of a linear frequency sweep over the given bandwidth (BW), encoding a symbol with the initial frequency. The symbol duration (SD), depending on the SF, can be obtained by $SD = \frac{2^{SF}}{BW}$.

A typical LoRa PHY frame consists of a preamble, a header and a payload. In the preamble, there are i) a series of symbols

with value 0, ii) a synchronization word, which is a network identification, and iii) two and a quarter down-chirps as an identifier of the end of the preamble. Except for the identifier of the end of the preamble, all the other parts use up-chirps. Figure 1 shows an example of a partial LoRa frame, which has a short series of one symbol with value 0, two symbols as the synchronization word, the identifier of the end of the preamble, and several symbols of data.

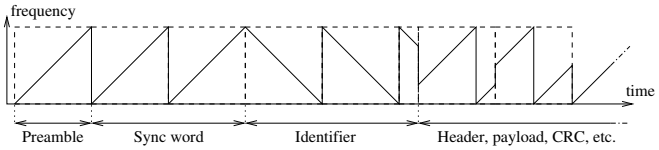

Fig. 1: Example of a partial LoRa frame.

To decode the symbols in a LoRa frame, the receiver synchronizes with the LoRa signal through the preambles. In each symbol duration, the received signals are multiplied by a normalized down-chirp to remove the time-variant. Then, the Fast Fourier Transformation (FFT) is performed on the production. The receiver finds the FFT peak in each symbol duration to get the symbol value. Finally, the decoded symbols compose an entire frame of the synchronized LoRa signal.

B. LoRa coding

The patented LoRa PHY [11] describes several coding techniques to enforce the robustness of the system. Authors in [12] have reverse-engineered LoRa and provided more details. An encoder transforms the data of a MAC frame into a LoRa frame, which consists of LoRa symbols. The coding techniques are combined into a pipeline, which consists of Hamming coding, whitening, interleaving, and Gray coding.

1) Hamming coding: Encoder adds redundancies to the frame for Error Correction Coding (ECC). The coding takes effect on each nybble of 4 bits from the MAC frame to generate codewords. In LoRa coding, a parameter named coding rate (CR), varying from 1 to 4, controls the number of redundant bits to add to each nybble. The redundant bits are derived from each nybble using Hamming coding. Several Hamming codings can be chosen in LoRa: 4/5, 4/6, 4/7, and 4/8, where 4 is the length of the nybble, and 5 to 8 is the length of the generated codewords, which equals $CR + 4$.

2) Whitening: An XOR operation with a specific sequence given by device manufacture is applied to the codewords after Hamming coding.

3) Interleaving: LoRa uses a diagonal interleaver to distribute the data bits over several symbols to distribute short-time noise or interference over several symbols. The interleaver acts on codeword blocks and scatters the bits of a codeword, from a line in the input matrix, to a diagonal in the output matrix [12]. The actual interleaver used in LoRa coding is reverse-engineered in [13]. It is applied to each block of SF codewords, where each codeword has $CR+4$ bits, to generate blocks of $CR + 4$ symbols, where each symbol has SF bits. A MAC frame may not be able to provide enough codewords

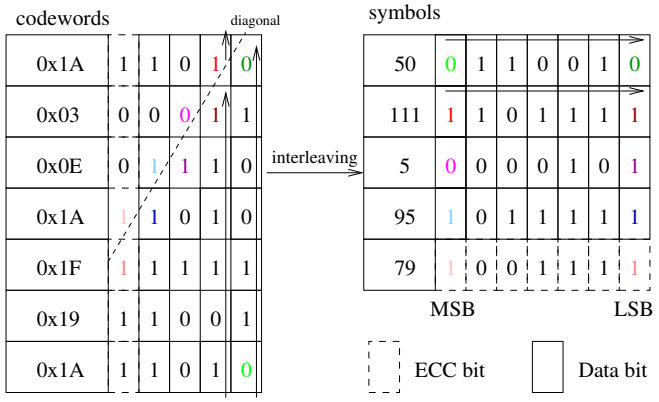

Fig. 2: Example of the diagonal interleaver applied to a codeword block (left) with SF = 7 and CR = 1, to get a symbol block (right).

to every block. In this case, the LoRa encoder needs to add paddings for encoding.

Let us consider the input matrix on the left of Fig. 2 as an example. Each codeword of the input matrix has one ECC bit (coming first) and four data bits (coming last). The bits forming each symbol in the output matrix start from the bit above the diagonal of the input matrix, and end at the bit below the diagonal, following the direction of the arrow in the input matrix. From the bottom to the top, the bits in the last column compose the first symbol 50 in the output matrix. For the penultimate column, the bits of the output symbol 111 are 1101111, as they start from the bit above the diagonal (row 1) and end at the bit below the diagonal (row 2).

4) Gray coding: Transmitters then encode symbols with Gray code, and receivers decode symbols vice versa.

C. LoRaWAN

LoRaWAN [14] is a simple MAC protocol on top of the LoRa physical layer. It defines a star topology, where end devices connect to a network server through gateways. End devices use LoRa to communicate with gateways. Gateways communicate with the network servers through IP connections.

LoRaWAN defines three classes of end devices. Class A is for low-power up-link communications and is mandatory. Class B provides a beacon-based mechanism for delay-guaranteed down-link communications. Class C does not have any limit on energy consumption. This paper concentrates on Class A, where the end devices can transmit at any time using the ALOHA mechanism: an end device chooses a random channel, sends a frame, and waits for potential frames from the gateway by opening two receive windows after transmission. The gateway can send the acknowledgment or the data from the network server during either of these receive windows. In confirmed traffic, retransmissions of frames can be triggered several times by end devices until receiving a valid acknowledgment or reaching the maximum number of transmissions allowed.

LoRaWAN provides different LoRa settings adapted to each region. European regional settings of LoRaWAN [4] define

seven data rates (DRs) for LoRa communication, from $DR0$ to $DR6$. The bandwidth of channels is 125 kHz for $DR0$ to $DR5$ and 250 kHz for $DR6$. SF varies from 7 to 12 for $DR5$ to $DR0$ and is equal to 7 for $DR6$. End devices can automatically adapt the data rate according to the quality of links. When an end device experiences a low signal quality, it decreases its DR to increase the transmission range and robustness of the signal. However, lower DR brings a lower bitrate. The indicative physical bit rate is 11000 bps for $DR6$ and down to 250 bps for $DR0$.

III. Existing collision resolution algorithms and uncertainties from the algorithms

This section first describes the existing collision resolution algorithms, CHOIR, SIC, GS-MAC, and the uncertainties produced by each one of these algorithms. Finally, we give an example of uncertainties in general.

A. CHOIR algorithm

CHOIR uses the standard LoRa decoding mechanism described in II-A to decode symbols. However, collided LoRa frames can become indistinguishable. In other words, the receiver cannot match the decoded symbols to the frames. To cope with this, authors in [5] proposes to leverage the small frequency offsets among transmitters to separate the symbols. The frequency offsets come from the oscillators of transmitters, due to the natural hardware imperfections.

In the core algorithm, the samples of the received signals are firstly multiplied by a normalized down-chirp. Then, the production is padded with 0 into a wider window (10× larger). Finally, they perform the FFT on the zero-padded production to extract the symbol values and the small frequency offsets from the signals. With a ten times larger FFT window, we can tell the resolution of the frequency offset is 0.1 over a symbol unit. For example, a demodulated symbol with value 51.1 (symbol 51 with an offset 0.1) and another symbol with value 61.2 (symbol 62 with an offset 0.2) are from different frames. When the offset between two frames is smaller than the resolution, these frames are still indistinguishable, and thus, uncertainties appear in some symbols upon decoding. The number of transmitters increases by scaling a LoRa network up, limiting the overall performance of the network.

B. SIC algorithm

Authors in [6] and [7] propose their algorithms based on SIC to decode two collided LoRa frames. The core mechanism is to leverage the capture effect in LoRa to decode the strongest signal, which needs to have significant differences from the other collided frames in collisions. The decoder synchronizes with the strongest LoRa signal to decode the entire LoRa frame through the standard LoRa decoding mechanism described in II-A. Then, the corresponding signal component is reconstructed and subtracted from the received signals. The decoder can then repeat the decoding process to decode the frame with the strongest signal from the residue of signals.

[6] also gives the sensitivity thresholds for signals with 125 kHz of bandwidth. For SF 7 and SF 12, the differences between the power of the strongest signal and the power of the collided signals should be, respectively, -6 dB and -20 dB. Otherwise, the algorithm can still decode symbols, but does not know the corresponding frames, which fails the algorithm and degrades the performance of the network.

C. GS-MAC algorithm

In GS-MAC, the gateway sends periodic beacons on each SF. Each beacon contains the number of slots before the next beacon. A slot starts with M sub-slots, equally distributed during the first symbol of the slot. The transmissions of end devices always start during the first symbol of a slot, in a random sub-slot. The gateway sends acknowledgments right after a correctly decoded transmission, in the same slot as the transmission. When there are any uncertainties during the decoding, the frame can not be fully decoded. The uncertainties occur because of the following three aspects.

1) Several collided frames in one sub-slot: When several end devices send their frames in the same sub-slot, GS-MAC can compute the possible symbols. However, a gateway does not know to which frame each symbol belongs.

2) Repeated symbols: GS-MAC is unable to detect a symbol change at the symbol frontier if two consecutive symbols are identical in one frame, which might cause uncertainties in the other collided frames.

3) Interference on first symbol: Based on the assumption in [8], the first symbol of frames can have uncertainties due to the interference between the up-chirps of the payload and the down-chirps of the end of preamble delimiters from other frames. [8] proposed to introduce an arbitrary symbol, which is different from the first data symbol to help decoding it.

D. Examples of Uncertainties

Suppose the two frames experience a collision with each other. In that case, we can have a synchronized case, in which the two frames are fully overlapped with each other from the beginning; or a desynchronized case, in which a frame arrives later than another one, overlapping a part of symbols.

F.1	50 111 5 95 79 26 33 94 124 18 52 27 58 116 94 1 0 32 16 0
F.2	80 95 88 29 104 67 31 81 45 91 98 9 19 10 99 1 64 32 16 8

TABLE I: EXAMPLE OF THE PAYLOAD OF TWO FRAMES (F.1 AND F.2) WITH SYMBOLS ENCODED BY SF 7

Let us consider two frames from two transmitters in Tab. I. If the two frames are synchronized, the receiver will be able to decode {50, 80} for the first symbol, {111, 95} for the second symbol, and so on[1]. However, the uncertainties appear here because sometimes it is impossible to match the symbols to each frame. For the cases where the two frames are partially overlapped, we show an example that the second frame arrives later than the first frame. The first symbol 80 in Frame 2 overlaps with the last symbol 0 in Frame 1, which produces

[1]This is the worst case which produces the uncertainties in the entire frame.

a single uncertainty with two possibilities {0, 80} for the last symbol in Frame 1, and the first symbol in Frame 2. SIC and CHOIR can conditionally resolve it. In GS-MAC, such a case is impossible because it has a slot-based synchronization mechanism. However, repeated symbols and the interference on the first symbol can incur such uncertainties in some symbols in a frame. Indeed, such uncertainties are much easier to remove because they produce fewer possibilities.

The receiver can conditionally resolve these cases by leveraging the minor frequency offsets in CHOIR, the significant power difference between the signals in SIC, or the different sub-slots in GS-MAC. However, the algorithms fail when certain conditions are not met, as described in each algorithm.

IV. Proposed Algorithm

In this section, we first describe the system model of our proposition. Second, we present our algorithm to reduce the uncertainties upon decoding LoRa frames. Finally, we give an analysis of our proposed algorithm.

A. System model

In LoRa, the total number of symbols N in a frame after LoRa coding is defined in [15] as follows:

$$N = 8 + \max\left(\lceil \frac{2PL - SF + 6 + 5H}{SF - 2LD}(CR + 4)\rceil, 0\right) \quad (1)$$

where PL is the number of bytes of the original data; SF is the spreading factor; H indicates if the LoRa PHY header is present (H=1) or not (H=0); LD indicates the low data rate mode (1 for enabled, 0 for disabling) in LoRa; and CR varies from 1 to 4, as the coding rate used by Hamming coding. Equation 1 can be simplified as follows:

$$N = 8 + (CR + 4)K, K \geq 0, \quad (2)$$

where K is the number of symbol blocks containing $(CR+4)$ symbols in each. The number K a function of modulation settings such as CR and SF.

With this representation, we can thus consider a LoRa frame as a concatenation of one symbol block as header, and K symbol blocks as payload. The symbol block of the header consists of eight symbols, encoding the payload length, CR for decoding the payload and the presence of CRC with $CR = 4$. Each symbol block in the payload consists of $CR+4$ symbols, as the output matrix in Figure. 2.

If we take the worst case between the two collided frames in Tab. I, the uncertainties can be represented by four symbol blocks with $CR = 1$ and $SF = 7$, as shown in Tab. II.

B. Decoding algorithm to reduce uncertainties

Our algorithm manipulates the uncertainties based on the symbols blocks from LoRa coding. Specially, during encoding, the interleaving process distributes the data bits and the redundant bits of codewords in a block among several symbols in a symbol block. By leveraging this property, we can significantly reduce the number of uncertainties in each block when decoding.

If there are any uncertainties in a symbol block, which are interpreted as several possible symbol values, the decoder generates the possibilities and deinterleaves them. It then calculates the expected redundant bits $calECC$ from the data bits. The encoded redundant bits $refECC$ should match $calECC$ for all codewords in a possible block. If this is not the case for one possible block, it can safely be removed. This process validates all the codewords, which are deinterleaved from blocks of symbols. Its detail is given in Algorithm 1.

Algorithm 1: Validation of codewords

```
S ← all possible symbol blocks from uncertainties
validatedBlocks ← {}
foreach block in S do
    codewordBlock ← deinterleave block
    flag ← true
    foreach codeword in codewordBlock do
        d ← extract data bits
        refECC ← extract ECC from data
        calECC ← calculate Hamming coding with d
        if refCRC ≠ calCRC then
            flag ← false
    if flag = true then
        add codewordBlock into validatedBlocks
```

Algorithm 2: Validation of CRC

```
foreach possible frame in decoded MAC frames do
    truncate frame according to the frame length
    refCRC ← retrieve CRC from MAC frame
    calCRC ← calculate CRC from MAC frame data
    if refCRC ≠ calCRC then
        drop the MAC frame
    else
        send the MAC frame to Network Server
```

Note that each block takes a position in the LoRa frame to which it belongs. The position does not change during interleaving and deinterleaving. Thus, after the validation of codewords, the data encoded in the codewords of each position can be extracted and assembled to get the possible MAC frames. The CRC of the payload is encoded into the last two bytes in a MAC frame. The CRC of the possible MAC frames can be calculated from the data in the MAC frames. By leveraging CRC, we can further reduce the possible MAC frames by comparing the CRC computed from the data with the CRC retrieved from the MAC frames. Algorithm 2 shows how the second step of our algorithm works.

We then show how the use of Algorithms 1 and 2 can reduce these uncertainties in an example as follows.

Let us consider the two frames in Tab. I, encoded with $SF = 7$ and $CR = 1$. There are uncertainties (generally, two) for each symbol in the frames. The gateway knows that

Block 1					Block 2				
Symbol 0	Symbol 1	Symbol 2	Symbol 3	Symbol 4	Symbol 5	Symbol 6	Symbol 7	Symbol 8	Symbol 9
{50, 80}	{95, 111}	{5, 88}	{29, 95}	{79, 104}	{26, 67}	{31, 33}	{81, 94}	{45, 124}	{18, 91}
Block 3					Block 4				
Symbol 10	Symbol 11	Symbol 12	Symbol 13	Symbol 14	Symbol 15	Symbol 16	Symbol 17	Symbol 18	Symbol 19
{52, 98}	{9, 27}	{19, 58}	{10, 116}	{94, 99}	{1}	{0, 64}	{32}	{16}	{0, 8}

TABLE II: EXAMPLE FOR UNCERTAINTIES FROM THE PAYLOAD OF TWO SUPERPOSED FRAMES, GROUPED BY BLOCKS

Validated codewords from Block 1	Validated codewords from Block 2	Validated codewords from Block 3	Validated codewords from Block 4
1. {1A 06 1A 1A 0B 0E 0D}	1. {19 1F 0E 12 12 1A 0D}	1. {14 12 03 14 0B 17 1C}	1. {0D 00 00 00 00 00 00}
2. {1A 06 02 1A 03 0E 0D}	2. {11 1F 06 1A 12 1A 0D}	2. {1C 1A 0B 14 03 1F 14}	2. {1F 00 00 00 00 00 00}
3. {1A 03 0E 1A 1F 19 1A}	3. {08 1F 08 05 05 1C 0E}	3. {00 03 14 14 1A 11 0D}	
4. {1A 03 06 1A 17 19 1A}	4. {00 1F 00 0D 05 1C 0E}	4. {08 0B 1C 14 12 19 05}	

TABLE III: EXAMPLE FOR REDUCED UNCERTAINTIES FROM TWO FRAMES LEVERAGING ECC IN CODEWORDS (IN HEX)

Frame	Symbol values of the validated LoRa frame, with SF 7 and CR1	Decoded MAC frame	CRC
1	50 111 5 95 79 26 33 94 124 18 52 27 58 116 94 1 0 32 16 0	0xA3 0xEA 0xF9 0xA8 0xF8 0x55 0xCE 0xCA 0xB4	0x3F 0x4D
2	80 95 88 29 104 67 31 81 45 91 98 9 19 10 99 1 64 32 16 8	0xA3 0x6A 0x79 0xA8 0xF8 0x55 0xCE 0x8B 0xC4	0x29 0x5F
3	50 111 5 29 79 26 33 94 124 18 98 9 19 116 99 1 64 32 16 8	0xA6 0x2A 0x3E 0xD1 0xF6 0xA2 0xAD 0x03 0x44	0xA1 0xDF

TABLE IV: EXAMPLE FOR THREE FRAMES, DECODED FROM THE TWO FRAMES USING PROPOSED ALGORITHM

every $CR + 4$ symbols compose a symbol block. We take the first five sets: {50, 80}, {111, 95}, {5, 88}, {95, 29}, and {79, 104} from Block 1 of frame 1 and frame 2, shown in Tab. II. By combining them, we get 32 possible blocks. Figure 3 shows two possible blocks: one with symbols {50, 111, 5, 95, 79}, and the other with symbols {50, 111, 88, 95, 79}. The only difference between these two blocks is the third symbol with a value of 5 or 88. After deinterleaving, we can see that the third column of codewords is 0010100 for symbol 5, and 0100011 (red) for symbol 88. Symbol 88 changes some codewords and the expected ECC, as shown in the ECC column. The expected ECC bits do not match the ECC bits in the codewords. Thus, the symbol block {50, 111, 88, 95, 79} can be removed from the possible block set, and the other symbol block {50, 111, 5, 95, 79} is considered as a possible block in a LoRa frame. Through Algorithm 1, the number of possible blocks for Block 1 is reduced from 32 to 4, as shown in Tab. III.

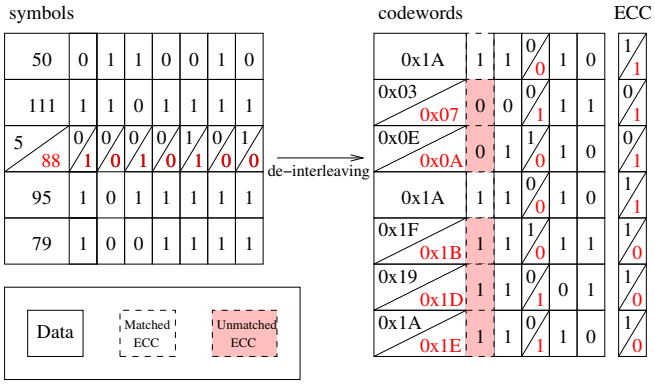

Fig. 3: Example of diagonal interleaver applied on a valid codeword block (left) and an invalid codeword block with SF = 7 and CR = 1.

Each possibility at each position in a frame contains SF codewords, which are deinterleaved from the corresponding symbol blocks. We can thus obtain 128 possible MAC frames (from 4 possibilities for Blocks 1, 2, 3, and 2 possibilities for Block 4) by extracting and concatenating the data in order. For example, if we take the first possibility for each position in Tab. III, which are {0x1A 0x06 0x1A 0x1A 0x0B 0x0E 0x0D} for Block 1, {0x19 0x1F 0x0E 0x12 0x12 0x1A 0x0D} for Block 2, {0x14 0x12 0x03 0x14 0x0B 0x17 0x1C} for Block 3, and {0x0D 0x00 0x00 0x00 0x00 0x00 0x00} for Block 4. By taking the lower 4 bits as data and remove the padding, we can get a frame {0xA6 0xAA 0xBE 0xD9 0xFE 0x22 0xAD 0x42 0x34} and the CRC {0xB7 0xCD} decoded from the frame. Algorithm 2 calculates the CRC from the MAC frame data and compares it with the CRC in the possible MAC frame, to further reduce the possibilities.

Finally, Table IV shows three frames that we decoded from the uncertainties with the help of Algorithms 1 and 2. We find that frame 1 and frame 2 are the frames transmitted by the end devices, according to Tab. I. Frame 3 is incorrectly decoded by our algorithm. However, the incorrectly decoded frames are rare when there are not many uncertainties. We will show this later in Subsection VI-B.

C. Complexity analysis

The frames in collision meet certain conditions and can produce several sets of indistinguishable frames. For example, the frames transmitted in sub-slot 1 in GS-MAC produce a set of indistinguishable frames, and the frames transmitted in sub-slot 2 produce another set. So, we can consider the complexity of each set of indistinguishable frames independently. Let us take only one set of C indistinguishable frames as an input.

To validate a codeword block, Algorithm 1 applies Hamming coding on the nybble of each codeword and compares it with the ECC in the codeword. The length of a nybble is fixed at 4. So, we can maintain a small dictionary of 16 different nybble values and their corresponding ECC for each CR. Thus, calculating Hamming coding becomes a query in a dictionary, which takes constant time. The complexity of validating a codeword block of SF codewords is $\mathcal{O}(SF)$. To

obtain a codeword block from a symbol block, the algorithm uses deinterleaving, which takes a $(CR+4) \times SF$ matrix and returns a $SF \times (CR+4)$ matrix. The manipulation is just a rearrangement of the data bits. So, its time complexity is $\mathcal{O}(SF.(CR+4))$, or $\mathcal{O}(SF.CR)$.

The time complexity on the validation of codewords is $\mathcal{O}(SF.CR.K.C^{CR+4} + 8.SF.C^8)$, where K is the number of blocks of the payload in a LoRa frame. When C is not large, the algorithm can decode many indistinguishable frames with acceptable computational overhead. For example, suppose we have $C = 2$, which means that two collided frames are not distinguishable by a collision resolution algorithm. In that case, the number of possible frames from the uncertainties can be up to $2^{8+K(CR+4)}$. Algorithm 1 performs the deinterleaving and the validations for each block with $CR+4$ symbols. The decoder can perform $8SF \times 2^8 + SF \times (CR+4) \times 2^{K(CR+4)}$ bit manipulations rather than iterating all the $2^{8+K(CR+4)}$ possible LoRa frames to firstly reduce the uncertainties. In our example, if we only consider the payload, the number of possible LoRa frames from the uncertainties in II is 2^{17} because the symbols 15, 17, 18 are identity for the two frames. However, we successfully reduce the number of 2^7 (around 0.1% remaining) by leveraging Algorithm 1.

After validating blocks, we can get some validated blocks, and we know their corresponding positions in possible LoRa frames. By concatenating data bits in the deinterleaved blocks, Algorithm 2 obtains a set of P possible MAC frames and their corresponding CRCs. We use L to denote the length of the largest frame. Calculating CRC takes linear time by iterating all the data bits in a frame. Thus, the time complexity of validation of CRC is $\mathcal{O}(P.L)$. The number of possible MAC frames, P, depends on the number of validated blocks from Algorithm 1. The number of validated blocks increases by increasing C, and decreases by increasing CR. Indeed, larger CR means there are more ECC bits in a codeword, which increases the probability of reducing the number of possible blocks. P only depends on the number of indistinguishable frames when CR is constant. In our example, P is obviously 2^7 with $CR = 1$. The payload has nine bytes and L is 9. As a result, we need to compute 128 CRC from 1152 bytes to get the final frames.

We can tell that the complexities of Algorithms 1 and 2 largely depend on C. It is possible to define a threshold for this number. The gateway does not need to perform extensive computations when the possible frames are numerous.

V. Simulation Results

In this section, we first present the parameter settings in our simulations, and then show the performance of our proposed algorithm, assuming uncertainties from different protocols.

A. Parameter settings

We implemented CHOIR, SIC, and GS-MAC using a LoRaWAN module [2] in the network simulator NS-3. We present the parameter settings in the following. The network consists of a network server, a single gateway, and several end devices uniformly distributed within one kilometer around the gateway, as shown in Fig. 4. We vary the number of end devices from 400 to 2000 for SF 7, and from 200 to 1000 for SF 12. CR is set to 1 (although larger values of CR would improve the performance of our algorithm).

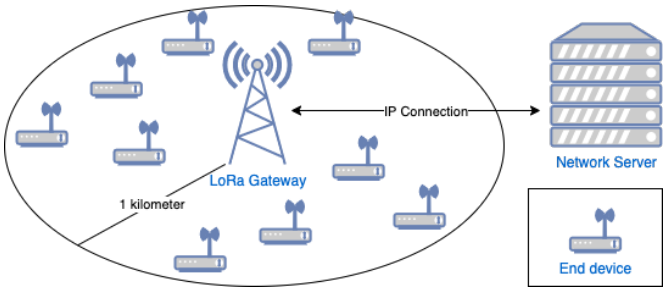

Fig. 4: Architecture of simulated LoRa network with several end devices, one LoRa gateway and one network server.

We simulate unconfirmed traffic during one hour. Each end device sends one frame with 10 bytes of random payload (after applying LoRa coding with CR 1, the frame corresponds to 28 symbols for SF 7, and to 38 symbols for SF 12). The gateway simulates an SX1301 hardware [16] with eight demodulation paths, which allows decoding up to eight frames simultaneously. We do not limit the decoding time nor the number of possible frames for our proposed algorithm.

We consider the European regional setting of LoRaWAN for all protocols. The bandwidth is set to 125 kHz. All devices use the same channel at 868.1 MHz in order to simulate a heavy traffic. The demodulation paths are also set to listen on the 868.1 MHz channel. We assume that the channel does not produce frame error or symbol error, thus the symbols are correctly demodulated by all algorithms. A log-distance path loss model is configured as the propagation model in the simulator. The loss can be represented as follows:

$$Loss = L_0 + 10n \log_{10} \frac{d}{d_0} \tag{3}$$

where $n = 3.76$, $L_0 = 7.7$ and $d_0 = 1$ are set.

The collision resolution algorithms and our proposed algorithm are only applied at the gateway.

- In CHOIR, end devices are initiated with uniformly distributed frequency offsets to simulate hardware imperfection. These frequency offsets are fixed during the whole simulation. We apply our algorithm when the collided frames have a relative frequency offset smaller than the given resolution, which is 0.1 symbol unit.
- In SIC, we consider that a large difference in signal power is sufficient to extract the frame with the strongest signal. [7] has shown that the error rate dramatically increases with three or four signal levels. So, we applied our algorithm only when several frames have similar signals that are the strongest, or the second strongest under only

[2] Available at https://github.com/signetlabdei/lorawan

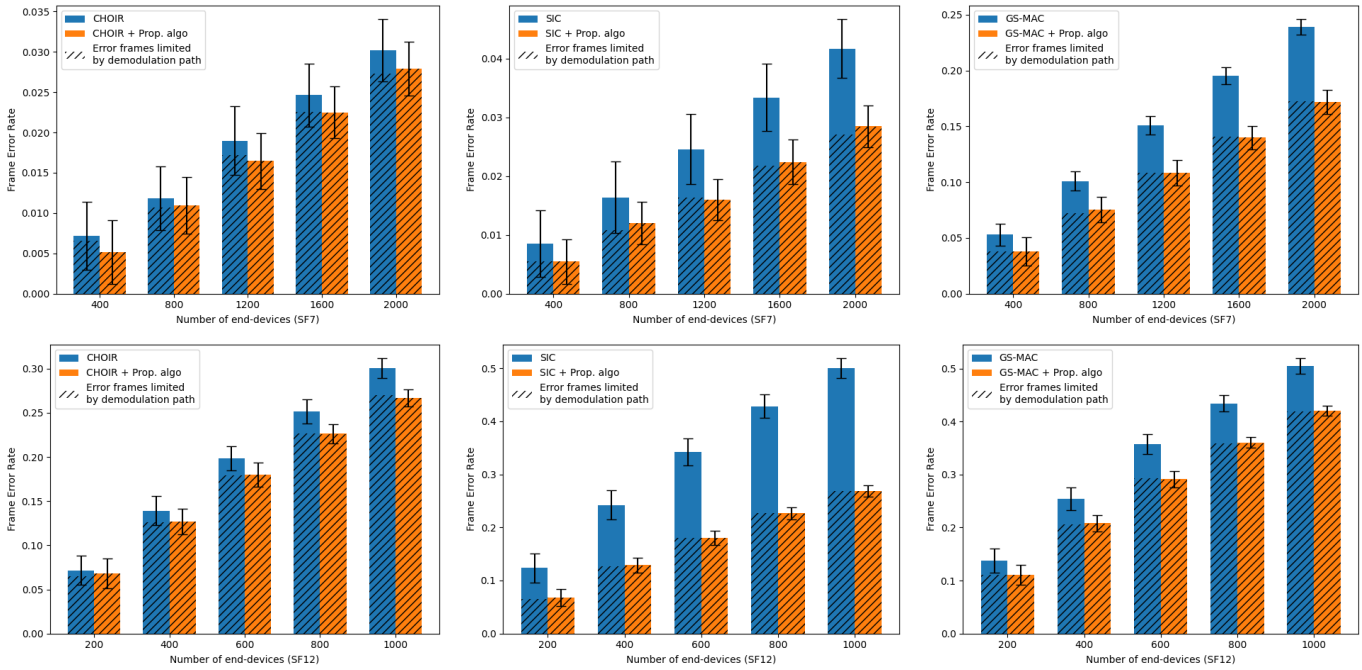

Fig. 5: Simulation results show that our proposed algorithm improves the decoding capacities for unconfirmed traffic with SF 7 and SF 12.

one stronger signal. The thresholds are set to -6 dB for SF 7, and -20 dB for SF 12, according to [6].

- In GS-MAC, the number of sub-slots is set to 4. The end devices randomly choose a sub-slot for each transmission, following a uniform distribution. Our algorithm is applied when there are uncertainties in a frame, regardless of the causes, that is, if several frames are sent during the same sub-slot, or if superposed symbols cause uncertainties.

The simulation results are averaged over 100 repetitions.

Note that the objective here is not to compare CHOIR, SIC, and GS-MAC, as each protocol benefit from different deployment settings, but rather to compare the benefits of our algorithm when applied to each protocol.

B. Frame error rate

We evaluate the decoding capacities in terms of frame error rate for all the frames sent during one hour. In the simulation, frame errors come either from collisions that the algorithms can not resolve, or by exceeding the limited number of demodulation paths (which is a hardware constraint of the SX1301).

Figure 5 shows the average frame error rate for each collision resolution algorithm, with and without our proposed algorithm, in terms of the number of end devices in the network. By adding our algorithm, we can tell that errors on frames are primarily due to the limited number of demodulation paths. Our algorithm reduces most of the uncertainties for SF 7 and SF 12, which brings fewer frame errors. With SF 7, compared to the original CHOIR, SIC, and GS-MAC, the gain with our proposed algorithm can reach, respectively, 13%, 35%, and 28%. With SF 12, the gain for each algorithm reaches up to 11%, 47%, and 19%, respectively.

As our simulation scenario deals with unconfirmed traffic, we did not show the results in terms of delay or energy efficiency, as they are slightly impacted by the decoding capabilities of the protocols. In the case of confirmed traffic however, reducing the frame error rate also reduces the number of retransmissions, which yields to a significant decrease in terms of both the delay and the energy consumption.

VI. Discussion

We briefly discuss here the limitations of our algorithm.

A. Computational overhead

The main limitation of the algorithm is the computation time required at the gateway. If many end devices simultaneously send frames, the computational overhead might be significant for the gateway, delaying the overall decoding, reception, and possibly acknowledgment. For example, for N indistinguishable frames, $CR + 4$ symbols can generate up to $N^{(CR+4)}$ possible blocks, which might bring too much overhead at the gateway. Thus, the gateway can set thresholds and waits for retransmissions if the number of indistinguishable frames exceeds the threshold.

B. Recovering fake frames

Our proposed algorithm might recover a few frames that have not been actually transmitted. Such an example is shown for frame 3 in Tab. IV. This occurs when a frame (here, frame 3) is completely composed of symbols from the other frames (here, frames 1 and 2), including a valid CRC. Since our algorithm generates all combinations of possible frames,

it generates frame 3, being a valid combination of frames 1 and 2, which might have been sent. Designing an algorithm that would minimize the number of possible frames is possible and would not generate frame 3, but such an algorithm would miss frame 3 if it was actually sent.

Such errors are theoretically impossible to avoid [9], without making assumptions on the upper layers. However, they can be detected and handled at a higher layer by the network server because the data is corrupted, which is out of the scope of this paper. The network server is not likely to reply to the fake frames, due to data corruption. Moreover, having the gateway acknowledge fake frames is not an issue, as either the end device would not exist, or the end device would not have opened the receive window for an acknowledgment (since it has not sent a frame).

Indistinguishable frames	2	4	6	8
Encoded with CR1, SF8	0.05%	2.82%	37.50%	91.12%
Encoded with CR2, SF8	0.09%	1.67%	16.11%	55.21%
Encoded with CR3, SF8	0.05%	0.07%	0.22%	0.88%
Encoded with CR4, SF8	0.05%	0.05%	0.20%	0.63%

TABLE V: PERCENTAGE OF FAKE FRAMES IN ALL DECODED FRAMES OVER 2000 REPETITIONS

In practice, this situation occurs very rarely when there are not many indistinguishable frames. Table V shows the statistical results of the percent of fake frames with different CRs and different numbers of indistinguishable frames. We can tell that as CR increases, there are fewer fake frames.

C. Potential performance issue

Both the CSS modulation and the LoRa coding enable LoRa to be robust to minor errors in symbols. Our algorithm currently assumes that all symbols are correctly received, confirmed in a scenario with a fair SNR and no significant time drift between transmitter and receiver. In the case where the SNR is low, additional uncertainties might come from erroneous symbol decoding. In this case, the performance of our algorithm might be degraded. The study of our algorithm in a noisy environment is left as future work.

VII. Conclusion

Collisions in LoRa negatively impact network performance. Many works are concentrating on decoding collided LoRa signals. However, there are specific cases where each protocol cannot decode all frames. In this paper, we proposed a general algorithm leveraging LoRa coding mechanisms to process indistinguishable frames. Simulation results showed that our algorithm can improve the decoding capabilities of most collision resolution algorithms. In our future work, we will explore how to cooperate with multiple gateways by separating the workload and leveraging the different characterizations of signals due to the relative positions of multiple gateways.

References

[1] Semtech Corporation. AN1200.22 LoRa Modulation Basics. Application note Revision 2, Semtech, 2015.

[2] Sigfox. http://www.sigfox.com. accessed 2020-10-01.

[3] Ingenu. http://www.ingenu.com. accessed 2020-10-01.

[4] Semtech Corporation. LoRaWAN v1.1 Regional Parameters. Technical Report Revision A, Semtech, 2017.

[5] R. Eletreby, D. Zhang, S. Kumar, and O. Yağan. Empowering Low-Power Wide Area Networks in Urban Settings. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, August 2017.

[6] B. Laporte-Fauret, M. A. Ben Temim, G. Ferre, D. Dallet, B. Minger, and L. Fuche. An Enhanced LoRa-Like Receiver for the Simultaneous Reception of Two Interfering Signals. In *Proceedings of Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, September 2019.

[7] M. A. Ben Temim, G. Ferre, R. Tajan, and B. Laporte-Fauret. A Novel Approach to Process the Multiple Reception of Non-Orthogonal LoRa-Like Signals. In *Proceedings of IEEE International Conference on Communications*, June 2020.

[8] N. El Rachkidy, A. Guitton, and M. Kaneko. Generalized Slotted MAC Protocol Exploiting LoRa Signal Collisions. In *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, September 2020.

[9] N. El Rachkidy, A. Guitton, and M. Kaneko. Decoding Superposed LoRa Signals. In *IEEE Conference on Local Computer Networks (LCN)*, October 2018.

[10] N. El Rachkidy, A. Guitton, and M. Kaneko. Collision Resolution Protocol for Delay and Energy Efficient LoRa Networks. *IEEE Transactions on Green Communications and Networking*, April 2019.

[11] O. B. A. Seller and N. Sornin. Low power long range transmitter, 2014.

[12] A. Marquet, N. Montavont, and G. Z. Papadopoulos. Towards an SDR implementation of LoRa: Reverse-engineering, demodulation strategies and assessment over Rayleigh channel. *IEEE Computer Communications*, March 2020.

[13] P. Robyns, P. Quax, W. Lamotte, and W. Thenaers. A Multi-Channel Software Decoder for the LoRa Modulation Scheme. In *Proceedings of International Conference on Internet of Things, Big Data and Security*, 2018.

[14] Semtech Corporation. LoRaWAN Specification v1.1. Technical Report Revision B, Semtech, 2017.

[15] Semtech Corporation. AN1200.13 SX1272/3/6/7/8: LoRa Modem Designer's Guide. Application note, Semtech, 2013.

[16] Semtech Corporation. SX1301 Datasheet v2.4 ed. Application note, Semtech, 2017.