Energy Efficient Data Recovery from Corrupted LoRa Frames

Yazdani, Niloofar; Kouvelas, Nikolaos; Venkatesha Prasad, R.; Lucani, Daniel E.

DOI
10.1109/GLOBECOM46510.2021.9685613

Publication date
2021

Document Version
Final published version

Published in
2021 IEEE Global Communications Conference (GLOBECOM)

Energy Efficient Data Recovery from Corrupted LoRa Frames

Niloofar Yazdani*†, Nikolaos Kouvelas†, R Venkatesha Prasad†, Daniel E. Lucani*
†Embedded and Networked Systems, Delft University of Technology, the Netherlands
*DIGIT and Department of Electrical and Computer Engineering, Aarhus University, Denmark
{n.yazdani,daniel.lucani}@ece.au.dk, {n.kouvelas,r.r.venkateshaprasad}@tudelft.nl

GLOBECOM 2021 - 2021 IEEE Global Communications Conference | 978-1-7281-8104-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/GLOBECOM46510.2021.9685613

***Abstract*—High frame-corruption is widely observed in Long Range Wide Area Networks (LoRaWAN) due to the coexistence with other networks in ISM bands and an Aloha-like MAC layer. LoRa's Forward Error Correction (FEC) mechanism is often insufficient to retrieve corrupted data. In fact, real-life measurements show that at least one-fourth of received transmissions are corrupted. When more frames are dropped, LoRa nodes usually switch over to higher spreading factors (SF), thus increasing transmission times and increasing the required energy. This paper introduces ReDCoS, a novel coding technique at the application layer that improves recovery of corrupted LoRa frames, thus reducing the overall transmission time and energy invested by LoRa nodes by several-fold. ReDCoS utilizes lightweight coding techniques to pre-encode the transmitted data. Therefore, the inbuilt Cyclic Redundancy Check (CRC) that follows is computed based on an already encoded data. At the receiver, we use both the CRC and the coded data to recover data from a corrupted frame beyond the built-in Error Correcting Code (ECC). We compare the performance of ReDCoS to (i) the standard FEC of vanilla-LoRaWAN, and to (ii) Reed Solomon (RS) coding applied as ECC to the data of LoRaWAN. The results indicated a 54x and 13.5x improvement of decoding ratio, respectively, when 20 data symbols were sent. Furthermore, we evaluated ReDCoS on-field using LoRa SX1261 transceivers showing that it outperformed RS-coding by factor of at least 2x (and up to 6x) in terms of the decoding ratio while consuming 38.5% less energy per correctly received transmission.**

***Index Terms*—LoRaWAN, Data recovery, Energy, CRC.**

I. INTRODUCTION

Numerous heterogeneous applications in Smart Cities and Industry 4.0 operate with devices that are usually energy constrained and require single-hop transmissions over large distances. To address the above need, the advances in RF-technology led to the rise of Low Power Wide Area Networks (LPWAN), providing energy efficient and long distance communications to IoT-devices. Long Range WAN (LoRaWAN) [1] is a popular LPWAN technology which – compared to others (e.g., NB-IoT, SigFox) – offers a "plug & play" approach, providing an easily accessible network. LoRa networks use sub-GHz license-free ISM bands, allowing their deployment without costs for spectrum usage. LoRaWAN enables inexpensive, and secure transmission of small payloads over large time intervals. The physical layer of LoRaWAN is based on LoRa and is proprietary, owned by Semtech. LoRa uses Chirp Spread Spectrum (CSS) mechanism. CSS enables long-range data transmissions and offers increased robustness to LoRa-signals, as they get detected even below the noise-level [2], [3]. Fig. 1 shows a typical LoRaWAN network, operating under a star of stars topology. As observed in Fig. 1, the data transmitted by the end-devices can be received by one or more gateways. The gateways forward the data – using the Internet – to the network server which is responsible for removing the duplicate messages and forwarding them to the application server. LoRa networks offer a wide range of configurable transmission possibilities – Spreading Factor (SF), transmission power, carrier frequency, channel bandwidth, forward error correction, and coding rate (CR). LoRa encodes each symbol, s, by $N = 2^{SF}$ *chips* and $SF \in 7, 8, ..., 12$. Accordingly, a symbol s is of length SF bits and $s \in \mathcal{S}$ where $\mathcal{S} = 0, 1, ..., N-1$ [4], [5]. Low SFs have high data rates – reaching up to 50 kbps– and relatively shorter transmission times but support shorter transmission ranges.

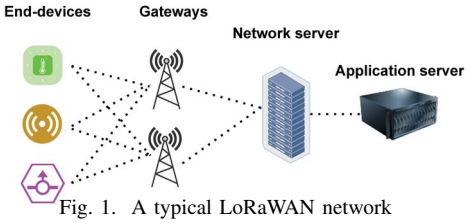

Fig. 1. A typical LoRaWAN network

A. Frame Corruption in LoRaWAN

Since the range of LoRaWAN is very high and the power limited because of the use of unlicensed band the Received Signal Strength Indicator (RSSI) is very low, *e.g.,* -115 dBm. Invariably the LoRa transmissions often suffer from symbol corruption including bursts of symbols. This renders LoRa-frames useless, and happens due to the following: (i) LoRa-frames can reach gateways located a few kilometers away; around 5 km in urban and 15 km in rural scenarios [6]. Over such long distances the corrupting effect of multipath propagation and physical phenomena, like shadowing and scattering, is predominant. In Non-Line of Sight (NLoS) conditions urban clutter intensifies the above phenomena. (ii) By operating in ISM bands LoRaWAN shares spectrum with other contending networks which interfere with LoRa-frames, and their transmissions corrupt LoRa-symbols. (iii) The Medium Access Control (MAC) layer of LoRaWAN is unslotted and Aloha-like, wherein every device transmits its data upon the generation without sensing the channel for any ongoing transmission. This aggravates further the issue of corrupted frames as it increases collisions.

In LoRaWAN, since there are no ACKs frame loss can lead to multiple issues. It leads to increased energy consumption due to retransmissions or in this case many more redundant frames or higher sampling rate to account for lost data points. Increased traffic may lead to more collisions and more data loss resulting in spiraling of higher energy consumption. Thus

978-1-7281-8104-2/21/$31.00 ©2021 IEEE

TABLE I
FRAME RECEPTION MEASUREMENT

payload size [B]	uncorrupted Received	Corrupted Received
14	35.24%	28.45%
18	45.28%	36.71%
24	16.54%	53.25%
28	23.93%	49.72%

the recovery of LoRa-transmissions is crucial to sustaining the battery life of the already constrained LoRa-devices as well as using the spectrum efficiently.

For data recovery, LoRa-modems use Forward Error Correction (FEC), and specifically Hamming codes [7], allowing CR of $\frac{4}{4+x}$ where $x \in \{1,2,3,4\}$. Namely, every 4 bit part of the frame is encoded into $4 + x$ bits. CR-values of 4/5 and 4/6 are capable only of error detection, while 4/7 and 4/8 can correct single errors or detect double errors. Uplink messages transmitted by end devices can use 4-bit Cyclic Redundancy Check (CRC) for their headers and 16-bit CRC at the end of their payloads for error detection. Moreover, the payload itself ends with 32-bit Message Integrity Code (MIC), i.e., a Cipher-based Message Authentication Code (CMAC) that assigns a specific signature to every end-device.

B. Constraints and Contributions

As observed through real-case experiments performed by Rahmadhani and Kuipers [8] and Marcelis *et al.* [9], the correction capability of the simple FEC mechanism of LoRa gets outperformed by the sheer numbers of corrupted symbols, which often lead to failed CRC-checks and dropped frames. Rahmadhani and Kuipers showed up to 32% of transmissions being corrupted in experiments wherein gateways were positioned 0.5-21.5 km away. Marcelis *et al.* observed up to 53% frame loss due to corruption when the range increases to 6 km

Additionally, we performed real-case experiments using LoRaWAN modules to investigate the level of corruption in LoRa-transmissions. Out of the 5000 frames transmitted per case, we found 29%-53% being received with corrupted symbols while using the lowest transmission power. The experiments took place in Line of Sight (LoS), with an operating frequency of 868.1 MHz, a bandwidth of 125 kHz on SF8 and SNR of around 11.50.

It is easy to see from the Table. I – which depicts the portion of corrupted frame-payloads in our experiments – that it is necessary to circumvent the frame corruption. Thus the question we try to answer here is: **Can we recover the data even if CRC check fails?**

However, any frame recovery algorithm needs to adhere to the constraints posed by LoRaWAN: (i) No changes should be made to the gateways and the existing infrastructure of LoRa networks. (ii) No ACK from the gateway is expected since it increases the message overhead and the energy consumption due to extra listening times. (iii) The introduced symbol redundancy due to encoding should be minimal, especially for high SFs, because the allowed payload size is limited (i.e., 51 B for SF10-SF12).

To this end, we introduce ReDCoS, Recovery of Data of Corrupted Signals; a novel coding scheme at the application layer of LoRaWAN. ReDCoS operates proactively before the LoRa-FEC mechanism is applied. Our contributions are the following: (i) We design ReDCoS, a novel application layer coding technique that allows the recovery of LoRa transmissions well beyond built-in error-correcting capability (see Sect. IV). (ii) We provide a mathematical analysis of ReDCoS, finding the probability of successful decoding (see Sect. V). (iii) We evaluate the performance of ReDCoS not only numerically but also experimentally, by conducting on-field experiments using LoRa-devices and a gateway (see Sect. VI).

II. RELATED WORKS

Marcelis *et al.* introduced DaRe, a coding scheme against data-loss, operating at the application layer of LoRaWAN. DaRe combines convolutional and fountain codes, providing 21% higher data recovery than LoRaWAN's repetition coding [9]. Coutaud and Tourancheau designed CCARR which uses Reed Solomon (RS) at the frame-level. CCARR transmits encoded super-frames, involving application data and redundancy data (RS-frames). CCARR guarantees high packet delivery ratio (even 100%) over lossy channels [10]. Borkotoky *et al.* propose two application-layer coding schemes –windowed and selective coding– for delay-intolerant LoRaWANs with minimum feedback. Windowed mechanism encodes by accounting all the non-delivered and non-expired transmitted symbols, while selective coding picks certain among them based on feedback [11]. Sant'Ana *et al.* combine two coding techniques of LPWANs: transmission of packet-replicas, RT, and transmission of encoded packets using linear operations like XORs, CT. Their hybrid coding scheme, which is optimized for LoRaWAN, can support the transmissions by more devices than RT and CT alone while extending their battery lifetime [12]. Chen *et al.* extend the lifetime and communication range of LoRaWAN, by introducing eLoRa, which recovers the correct parts of a packet by applying Luby codes (LT) on several versions of the packet that are received by multiple gateways. Further, eLoRa optimizes physical layer parameters to best serve lifetime and range requirements [13].

III. BACKGROUND

Cyclic Redundancy Check (CRC) are error-detecting codes used by communication and storage systems to detect corruption of data for instance caused by noise. To encode the data, considering the systematic approach, CRC-m appends a check value of a fixed length of m bits to the data using a hash function to map the data values to the m bits. Upon retrieval, if the data is corrupted the calculated CRC value does not match the appended value, thus a CRC error occurs.

Reed-Solomon codes (RS) [14] are a family of linear Error-Correcting Codes (ECC) used to correct errors in many applications including communication systems and storage systems. RS codes treat a data block, i.e., *message*, as a series of symbols from a finite field of size q, $\mathbb{F}_q$. A RS code is specified as $RS_q(n,k)$ where q denotes the finite field whose symbols are used, and k, n are the symbol-size of the message and the codeword, respectively, where $k < n$. According to the above, upon the arrival of a message of length k symbols, $\{x_1, x_2, ..., x_k\}$ where $0 \leq x_i < q$, the encoder generates a codeword of length n symbols, $\{y_1, y_2, ..., y_n\}$ where $0 \leq y_i < q$. The codeword length, n, is equal to $q - 1$; However, using a short version of RS codes [15], n is more

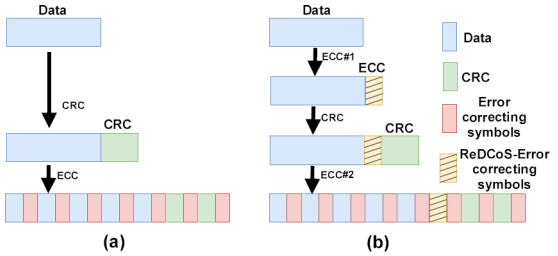

Fig. 2. The order of using a CRC and an ECC (a) Conventional method, (b) proposed scheme: ReDCoS.

flexible, *i.e.*, $k < n < q$. Let $t = n - k$, a Reed-Solomon code can correct up to and including $\lfloor \frac{t}{2} \rfloor$ corrupted symbols at unknown locations. An RS code can also be used as an *erasure code* where it can correct up to and including t erasures at **known locations**. In this paper, we use the functionality of RS codes as an erasure code. The encoding procedure of RS codes can be *systematic*, wherein messages appear at the beginning of the codeword followed by t error correcting symbols.

IV. Proposed Scheme: Data Recovery Beyond Built-In Error-Correction

In a typical wireless system, *e.g., WiFi*, the sender typically takes the data, computes the CRC, and applies an ECC, *e.g.*, Reed-Solomon or Hamming, on the data plus the CRC as shown in Fig. 2. (a). Note that in this paper, we consider using a systematic encoding procedure for the sake of simplicity. On retrieval, the receiver applies the decoding function of the ECC to retrieve the data and the CRC and to correct possible corrupted bits/symbols. Afterwards, the receiver calculates the CRC for the recovered data. If the calculated CRC matches the received CRC, data is received successfully. Otherwise, data cannot be retrieved and this happens if the number of corrupted symbols exceeds the correction capability of the applied ECC. We try to recover the data even if CRC check fails.

The Idea. The idea is to make the system more robust by adding a limited number of extra symbols using an ECC as shown in Fig. 2. (b). This redundancy is used later to create the correct CRC and the correct data using majority voting. The recovered CRC is later verified using parts of the received CRC that are not damaged. The important technique here is to **combine the concepts of the CRC and ECCs to utilize ECCs beyond their error-correcting capability.**

Encoder. Encoder is lightweight and suitable for resource limited devices, as it only appends a few extra symbols to the end of the data using the encoding function of an ECC (ECC#1, referred to as ReDCoS' ECC, hereafter). Such ECCs can be RS codes, as shown with yellow color (dashed area) in Fig. 2. (b). The rest of the process is similar to a typical wireless system where a CRC is calculated for the encoded data, and the encoding function of an ECC (ECC#2) is applied to the encoded data plus the CRC as shown in Fig. 2. (b). The two ECCs are applied independently. Note that by *encoded data*, hereafter, we refer to the data plus the few extra symbols added by the ReDCoS' ECC. In this paper, we consider CRC-32. We define l as the symbol-size of the CRC.

Decoder. When a corrupted frame is received the decoding of second ECC is done to recover the encoded data plus the CRC. The formal decoding algorithm for a given received payload is detailed in algorithm 1, where k denotes the symbol size of the data and t denotes the symbol size of the error correcting symbols added by ReDCoS' ECC. We also use $C(z,p)$ to represent the number of p-combinations from a given set of z elements where $C(z,p) = \frac{z!}{p!(z-p)!}$. The parameter H denotes the minimum number of required symbols matching the recovered CRC and the received CRC to verify the recovered CRC. According to algorithm 1, ReDCoS recovers the data if one of the following cases happens: (i) The received encoded data and the received CRC are both uncorrupted. (ii) The received CRC is uncorrupted and there is a minimum of k correct symbols out of the $k+t$ symbols of the received encoded farmes. (iii) A minimum of H symbols of the received CRC are uncorrupted (to verify the recovered CRC) and a minimum of $k+1$ symbols of the received encoded data are uncorrupted. In the last case, the recovered uncorrupted CRC will be repeated a minimum of $k+1$ ($C(k+1,k)$) times by trying different k-combinations from the given $k+t$ symbols for the encoded data. There are a total of $C(k+t,k)$ k-combinations that a decoder can try in brute force. *Majority voting*, in this paper, denotes finding CRCs which are repeated a minimum of $k+1$ times. Selecting the k-combinations could happen at random; however, a given k-combination should not be tried two times. By selecting a k-combination, the decoder calculates the remaining t symbols (that were not picked randomly by the decoder) for the uncorrupted encoded data candidate using the decoding and encoding functions of the ReDCoS' ECC.

In this paper, particularly, we concentrate on: (i) the encoded

Algorithm 1: ReDCoS' decoder:

Input: da_r: received data, ecs_r: received error correcting symbols, CRC_r: received CRC, H: minimum number of CRC symbol matches
Output: da_u: uncorrupted data, failed decoding
$CRC\_t \leftarrow \{\}$ // pairs of (CRC:#repetition)
Step 1: Check if the received encoded data is uncorrupted.
$i \leftarrow find\_CRC(da\_r, ecs\_r)$
if $i == CRC\_r$ **then**
  $return(da\_u = da\_r)$ // End: SUCCESS
for $j = \{1, 2, ..., C(k+t,k)\}$ **do**
  Step 2: Generate an uncorrupted encoded data candidate: Try a k-combination
  $y \leftarrow \{\}.size(k+t)$
  $y \leftarrow select\_k\_symbols(da\_r, ecs\_r)$
  $y \leftarrow calculate\_remaining\_t\_symbols(y)$
  $i \leftarrow find\_CRC(y)$
  Step 3: Check if the candidate matches the received CRC
  if $i == CRC\_r$ **then**
    $da\_u = pick\_first\_k\_symbols(y)$
    $return(da\_u)$ // End: SUCCESS
  Step 4: Save/Check the recovered CRC.
    // Check key i availability in CRC table CRC_t
  if $!find\_key(i, CRC\_t)$ **then**
    $CRC\_t[i] = 1$
  else
    $CRC\_t[i] += 1$
    Step 5: Majority voting.
    if $CRC\_t[i] == k+1$ **then**
      $x \leftarrow find\_\#similar\_symbols(i, CRC\_r)$
      Step 6: Verify the recovered CRC.
      if $x \geq H$ **then**
        $da\_u = pick\_first\_k\_symbols(y)$
        $return(da\_u)$ // End: SUCCESS
Step 7: return failure if algorithm has not ended yet by a return
$return(\text{"}Failed\ Decoding\text{"})$ // End: Failed

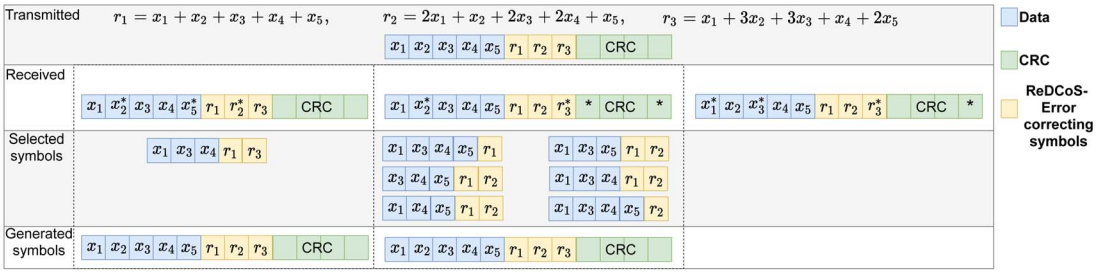

Fig. 3. Data retrieval examples, a) Success using the received uncorrupted CRC, b) Success using the partially uncorrupted received CRC. c) Failure.

data plus the CRC, *i.e.,* before applying the encoding function of the second ECC at the sender side, and (ii) the received encoded data plus the CRC, *i.e.,* after applying the decoding function of the second ECC at the receiver side.

Example 1. *Fig. 3 illustrates* 3 *examples for ReDCoS' decoder, including* 2 *successful decoding and* 1 *failed decoding. Let us take the data with* 5 *symbols* $\{x_1, x_2, x_3, x_4, x_5\}$*, i.e.,* $k = 5$*. The data is encoded as* $\{x_1, x_2, x_3, x_4, x_5, r_1, r_2, r_3\}$ *where* $t = 3$ *and* r_1, r_2*, and* r_3 *are independent linear combinations of* x_1, x_2, x_3, x_4*, and* x_5 *(the operations are over a given finite field). We assume that some of the symbols are corrupted while receiving. The corrupted symbols are shown with a star. The number of initially corrupted symbols could be higher but some are already corrected by the second ECC.*

In particular, we consider 3 scenarios:

Example1.1. 5 symbols out of $k+t$ symbols are uncorrupted while $5 = k$. While decoder is trying different k-combinations, as soon as it selects the error-free symbols, *i.e.,* x_1, x_3, x_4, r_1, r_3, it will construct the correct encoded data and hence it generates the correct CRC. As the received CRC is not corrupted, the generated CRC would be identical to the received CRC and data is retrieved successfully.

Example1.2. 2 symbols out of $k + t = 8$ symbols are corrupted. Furthermore, 2 symbols of the CRC are corrupted as shown in Fig. 3. (b). This means that 6 symbols out of $k + t$ symbols are uncorrupted where $6 = k + 1$ and 2 symbols of the CRC are uncorrupted as well (considering a 4-symbol CRC). The decoder tries different k-combinations which will lead to different CRC values. A total of $C(8,5) = 56$ different combinations exist. However, every time the receiver selects $k = 5$ symbols out of the 6 correct symbols, *i.e.*, $x_1, x_3, x_4, x_5, r_1, r_2$, the correct data and hence the correct CRC will be generated. This means that the correct CRC will be repeated $C(6,5) = 6$ $(k + 1)$ times with choosing the combinations shown in Fig. 3. (b). Verifying the recovered CRC, 2 symbols of the recovered CRC match the received CRC. Considering $H = 2$, the recovered CRC is verified and data can be retrieved successfully.

Example1.3. 3 symbols out of $k + t = 8$ symbols are corrupted in addition to 1 symbol of the CRC. This means that 5 symbols out of $k + t = 8$ symbols are uncorrupted, where $5 = k$. The only combination which generates the correct CRC is x_2, x_4, x_5, r_1, r_2. Thus, the correct CRC will be repeated only once similar to many other CRCs. Since the received CRC is also corrupted, there is no reliable way to retrieve the data.

V. Mathematical Analysis of ReDCoS

We evaluate ReDCoS in terms of **encoder and decoder throughput, decoding ratio, and false decoding ratio** for different Symbol Error Rates (SER). SER is the ratio of corrupted symbols over transmitted symbols, and it denotes the probability of a symbol getting corrupted during the transmission. The encoder and decoder *throughput* indicate the amount of data (i.e., excluding error-correcting symbols or CRC) processed. We define the *Decoding Ratio (DR)* as the ratio of decoded over total received payloads, and the *False Decoding Ratio (FDR)* as the ratio of falsely decoded over total decoded payloads. False decoding occurs if the recovered payload is not identical to the associated transmitted payload. Increasing H reduces the FDR as the recovered CRC must be validated more strictly, i.e., requiring more symbol-matches.

Decoding Probability. Without loss of generality, each symbol can get corrupted during transmission independently of other symbols under probability SER. $P(w,e)$ is the probability to have e corrupted out of w given symbols,

$$P(w,e) = SER^e \cdot (1 - SER)^{w-e} \cdot C(w,e), \forall e \in [0,w] \quad (1)$$

Received CRC is uncorrupted. The probability of receiving an uncorrupted CRC is $P(l,0)$. In this case, ReDCoS recovers the data if up to and including t symbols are corrupted out of the $k+t$ symbols. This happens with probability $\Sigma_{i=0}^{t} P(k+t,i)$.

Matched symbols between recovered and received CRC greater or equal to H and less than l. The number of corrupted symbols of the CRC is within $\{1,2,...,l-H\}$. This happens with probability $\Sigma_{i=1}^{l-H} P(l,i)$. In this case, ReDCoS recovers the data if up to and including $t-1$ symbols are corrupted out of the $k+t$ symbols (equivalently, $k+1$ symbols are intact). This happens with probability $\Sigma_{i=0}^{t-1} P(k+t,i)$.

Matched symbols between recovered and received CRC less than H. In this case, ReDCoS cannot decode the data. Based on the specific probabilities of each case above, the probability of successfully decoding a frame is,

$$\mathbb{P}_D = P(l,0) \cdot \Sigma_{i=0}^{t} P(k+t,i) + \Sigma_{i=1}^{l-H} P(l,i) \cdot \Sigma_{i=0}^{t-1} P(k+t,i) \quad (2)$$

VI. Performance Evaluation

We consider $RS_{256}(k+t,k)$ as the ReDCoS' ECC for evaluation. Moreover, all LoRaWAN frames are transmitted at SF8. Thus, each symbol is 1 byte. We compare the decoding ratio of the ReDCoS to two other approaches: (i) using Reed-Solomon codes as an ECC (RS-ECC) with the same number of error correcting symbols and data symbols, which is capable of data recovery if up to and including $\lfloor \frac{t}{2} \rfloor$ symbols of encoded data are corrupted and CRC is uncorrupted, and (ii) without using any added ECC (No-Added-ECC) with the same number

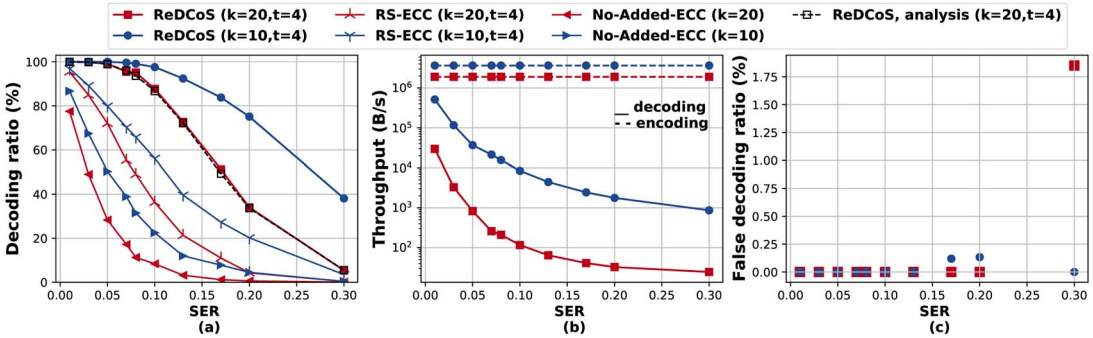

Fig. 4. (a) Decoding ratio, (b) encoder's and decoder's throughput, (c) false decoding ratio, all versus SER.

TABLE II
IMPLEMENTATION RESULTS AND CHARACTERISTICS

(k, t)	SNR(min, max, ave)	RSSI(min, max, ave)[dBm]	Received$_1$ frames	Corrupted$_2$ frames	encoder throughput[MB/s]	decoder throughput	encoder energy [nJ/b]
(10, 4)	(−14, −9, −11.9)	(−116, −114, −115.1)	63.7%	44.67%	0.5724	2087.7 B/s	38.6
(10, 8)	(−14, −1, −11.18)	(−116, −113, −115.0)	82.0%	44.77%	0.5348	128.6 B/s	41.3
(20, 4)	(−15, −2, −11.78)	(−116, −112, −115.1)	69.8%	76.3%	0.1580	39.3 B/s	140.0

[1] Uncorrupted and corrupted received frames. [2] # of corrupted received frames / # of received frames.

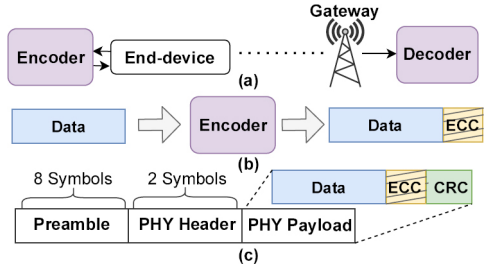

Fig. 5. Implementation. (a) LoRa device and a Gateway. (b) The encoder is implemented as a C++ library to compute error correcting symbols. (c) LoRaWAN frame. Note that there is no change in the LoRa frame format.

of data symbols, wherein data recovery is possible if all k data symbols and the received CRC are uncorrupted. This corresponds to vanilla-LoRaWAN.

A. Numerical Results

We use a Core i7-7820HQ at 2.90GHz for the tests and focus on a single thread (and CPU) for execution. We consider $H = 2$. Fig. 4 shows the ReDCos' performance for 2 different cases: $(k = 10, t = 4)$, and $(k = 20, t = 4)$ over 10 different values for SER. The measurements average over 1000 runs per case per SER. It is assumed that all the transmitted frames are received by the receiver. Fig. 4. (a) depicts the decoding ratio to SER. In general, as the values of SER increase the decoding ratio decreases for all 3 schemes. As shown in Fig. 4. (a), for $(k = 20, t = 4)$, ReDCoS outperforms RS-ECC and No-Added-ECC in terms of decoding ratio by up to 13.5 and 54.0 times, respectively. Moreover, for $(k = 10, t = 4)$, ReDCoS outperforms by up to 10.8 and 95.2 times, correspondingly. As depicted in Fig. 4. (b), ReDCoS' encoding throughput is 1.90 MBps and 3.65 MBps for ReDCoS$(k = 20, t = 4)$ and ReDCoS$(k = 10, t = 4)$, respectively. As shown in Fig. 4. (b), ReDCoS' decoding throughput decreases with the increase of SER because the number of corrupted symbols increases. The decoding throughput for ReDCoS$(k = 20, t = 4)$ ranges from 29.6 kBps for SER=0.01 to 24.4 Bps for SER=0.3. The decoding throughput for ReDCoS$(k = 10, t = 4)$ ranges from 513.7 kBps for SER=0.01 to 865.9 Bps for SER=0.3. Fig. 4. (c) shows FDR regarding ten different SER values. For ReDCoS $(k = 20, t = 4)$ and $(k = 10, t = 4)$, $FDR = 0$ for nine and eight out of ten cases, respectively. SER value can be decreased by increasing H.

B. Practical Evaluation

For our experiments, we consider one end-device (sender) and one gateway (receiver) as shown in Fig. 5. (a). We used LoRa SX1261 nodes, integrated with a Raspberry Pi 3 Model B. The sender and receiver antennas are isotropic with 0 dBi gain. The operating frequency and bandwidth are 868.1 MHz and 125 kHz, respectively. The encoder is implemented as a C++ library that is included in the end-device software. The encoder operates on each data and generates the encoded data as shown in Fig. 5. (b). Data contain random symbols. Fig. 5. (c) shows the transmitted LoRaWAN frame structure. We consider only the mandatory parts of LoRa-frames to keep the frame structure as simple as possible. The optional frame integrity checks and the code rate are deactivated to facilitate the reception of more corrupted frames during our experiments. Alternatively, a 4-byte CRC check associated with the encoded data is used as shown in Fig. 5. (c).

General. We evaluated 3 cases as shown in Table. II: $(k = 10, t = 4)$, $(k = 10, t = 8)$, and $(k = 20, t = 4)$. For each case, 5000 frames are transmitted. As depicted in Table. II, the SNR (Signal-to-Noise Ratio) value and the RSSI (Received Signal Strength Indicator) value of the received frames lies between -15 to -1 and between -116dBm to -112dBm, respectively. Best-case scenario, 82% of the transmitted frames are received while this value is 63.7% for $(k = 10, t = 4)$. Up to 76.3% of the received frames are corrupted. The decoder throughput is 2087.7 B/s for $(k = 10, t = 4)$. Increasing k or t increases the average number of possible iterations for finding the correct data and hence decreases the throughput. As depicted in Table. II, the throughput for $(k = 20, t = 4)$ is as low as 39.3 B/s. Contrarily, the encoder is lightweight. For instance, the encoder's throughput and energy consumption are 0.5724 MB/s and 38.6 nJ/s, respectively for $(k = 10, t = 4)$.

Decoding and False Decoding Ratios. Fig. 6 depicts the decoding ratio for ReDCoS and RS-ECC. This decoding ratio

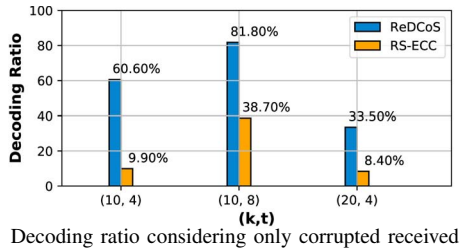

Fig. 6. Decoding ratio considering only corrupted received frames.

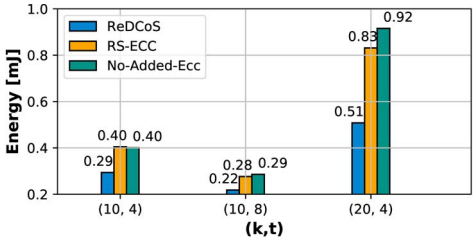

Fig. 7. Transmission energy for one correctly received frame.

is calculated by considering only received corrupted frames, *i.e.,* the decoding ratio shows the percentage of the received corrupted frames that are recovered successfully. As depicted in Fig. 6, ReDCoS outperforms RS-ECC in terms of decoding ratio by factors of 6.12, 2.11, and 3.99 times for (k = 10, t = 4), (k = 10, t = 8), and (k = 20, t = 4), respectively. Table. III, shows the portion of the received frames (corrupted and uncorrupted), and the portion of correctly received frames by the three approaches as well as the ReDCoS' FDR. As shown in Table. III, FDR is as low as 0%, 0.13%, 0.11% for the same above cases, respectively.

Energy: The main source of energy consumption in an end-device is data transmission, which can be calculated by multiplying the transmission power (0 dBm or 0.001 W) by the frame Time on Air (ToA). ToA depends on the SF, and bandwidth used [4]. However, if a frame is not correctly received, it needs to be retransmitted or it had to be coded over multiple frames [9]. To consider the effect of frame loss, the energy consumption per correctly received frame has been considered (see Table. III) and is shown in Fig. 7. As seen, ReDCoS consumes up to 38.5% and 44.5% less energy compared to ECC-RS and No-Added-ECC, respectively.

VII. Conclusions

LoRa offers a low bitrate transmission of IoT data. However, the MAC layer offers no guarantee and corrupted frames are dropped. In this paper, we proposed Recovery of Data of Corrupted Signals for the application layer of LoRa networks, wherein LoRa-payloads are pre-encoded using lightweight coding scheme before the addition of the CRC. This leads to more robust encoded packets after applying LoRa-FEC.

We provided the probabilistic analysis of successful decoding using ReDCoS. Further, we conducted numerical simulation and on-field evaluation of ReDCoS. The results show that ReDCoS outperforms manifolds not only the built-in error-correction scheme of LoRaWAN, but also RS codes when used as ECC. Additionally, because of ReDCoS' data-recovery capability, less packet retransmissions (or encoding over multiple frames) occur, reducing the energy consumption by 44.5% as well as age of information while doubling the frame reception. ReDCoS acts independently of the standard LoRa-encoding requiring no change to infrastructure and/or protocol. Moreover, it benefits from a simple encoder. Further, ReDCoS does not need buffer and/or feedback downlinks for its operation, keeping message-overhead low.

TABLE III
Measurements

(k, t)	Received frames	Correctly received, No-Added -ECC	Correctly received, RS-ECC	Correctly received, ReDCoS	FDR
(10, 4)	63.7%	35.94%	38.08%	52.48%	0%
(10, 8)	82.0%	50.36%	59.49%	75.31%	0.13%
(20, 4)	69.8%	17.96%	21.00%	34.37%	0.11%

All items are calculated considering all the transmitted frames.

VIII. Acknowledgement

The authors would like to thank Sujay Narayana from TU Delft for providing insight for doing the experiments.

This work was supported in part by the SCALE-IoT Project (Grant No. 7026-00042B) granted by the Danish Council for Independent Research, the Aarhus Universitets Forskningsfond Starting Grant Project AUFF- 2017-FLS- 7-1.

InSecTT has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876038. The JU receives support from the European Union's Horizon 2020 research and innovation program and Austria, Sweden, Spain, Italy, France, Portugal, Ireland, Finland, Slovenia, Poland, Netherlands, Turkey. The document reflects the author's view only and the Commission is not responsible for any use that may be made of the information it contains.

References

[1] N. Sornin, M. Luis, T. Eirich, T. Kramp, and O. Hersent, "LoRaWAN specification," *LoRa alliance*, 2015.
[2] J. Petäjäjärvi, K. Mikhaylov, M. Pettissalo, J. Janhunen, and J. Iinatti, "Performance of a low-power wide-area network based on LoRa technology: Doppler robustness, scalability, and coverage," *IJDSN*, vol. 13, no. 3, 2017.
[3] J. C. Liando, A. Gamage, A. W. Tengourtius, and M. Li, "Known and Unknown Facts of LoRa: Experiences from a Large-Scale Measurement Study," *ACM Trans. Sen. Netw.*, vol. 15, no. 2, 2019.
[4] Semtech, "LoRa® and LoRaWAN®: A Technical Overview," [Online; accessed: 2021-12-4].
[5] LoRa™ Alliance, "LoRaWAN® Specification v1.1," [Online; accessed: 2020-10-10].
[6] 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*, vol. 23, no. 5, pp. 60–67, 2016.
[7] T. Elshabrawy and J. Robert, "Evaluation of the BER Performance of LoRa Communication using BICM Decoding," in *IEEE ICCE*, 2019, pp. 162–167.
[8] A. Rahmadhani and F. Kuipers, "When LoRaWAN Frames Collide," ser. WiNTECH '18. ACM, 2018, p. 89–97.
[9] P. Marcelis, N. Kouvelas, V. S. Rao, and V. Prasad, "DaRe: Data Recovery through Application Layer Coding for LoRaWAN," *IEEE TMC*, 2020.
[10] U. Coutaud and B. Tourancheau, "Channel Coding for Better QoS in LoRa Networks," in *IEEE WiMob*, 2018, pp. 1–9.
[11] S. S. Borkotoky, U. Schilcher, and C. Raffelsberger, "Application-Layer Coding with Intermittent Feedback Under Delay and Duty-Cycle Constraints," in *IEEE ICC*, 2020, pp. 1–6.
[12] J. M. d. S. Sant'Ana, A. Hoeller, R. D. Souza, S. Montejo-Sánchez, H. Alves, and M. d. Noronha-Neto, "Hybrid Coded Replication in LoRa Networks," *IEEE TII*, vol. 16, no. 8, pp. 5577–5585, 2020.
[13] G. Chen, J. Lv, and W. Dong, "Exploiting Rateless Codes and Cross-Layer Optimization for Low-Power Wide-Area Networks," in *IEEE/ACM IWQoS*, 2020, pp. 1–9.
[14] S. B. Wicker and V. K. Bhargava, *Reed-Solomon codes and their applications*. John Wiley & Sons, 1999.
[15] J. C. Moreira and P. G. Farrell, *Essentials of error-control coding*. John Wiley & Sons, 2006.