This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3016654, IEEE Transactions on Mobile Computing

DaRe: Data Recovery through Application Layer Coding for LoRaWAN

P. J. Marcelis, V. S. Rao, *Member, IEEE*, N. Kouvelas, *Student Member, IEEE*, R. V. Prasad, *Senior Member, IEEE*

Abstract—Long-Range Wide-Area Network (LoRaWAN) is an energy-efficient and inexpensive networking technology that is rapidly being adopted for many Internet-of-Things applications. In this study, we perform extensive measurements on a new LoRaWAN deployment to characterise the spatio-temporal properties of the LoRaWAN channel. Our experiments reveal that LoRaWAN frames are mostly lost due to the channel effects, which are adverse when the end-devices are mobile. The frame losses are up to 70%, which can be bursty for both mobile and stationary scenarios.

Frame losses result in data losses since the frames are transmitted only once in the basic configuration. To reduce the data losses in LoRaWAN, we design a novel coding scheme for data recovery called DaRe that works on the application layer. DaRe combines techniques from convolutional and fountain codes.

By implementing DaRe, we show that 99% of the data can be recovered with a code rate of 1/2 when the frame loss is up to 40%. Compared to the repetition coding scheme, DaRe provides 21% higher data recovery and can save up to 42% of the energy consumed on a transmission for 10-byte data units. We also show that DaRe provides better resilience to bursty frame losses.

Index Terms—LoRaWAN, LPWAN, network measurements, forward error correction, data recovery, erasure coding, application layer coding, fountain codes, convolutional codes

1 Introduction

The miniaturisation of computing and communication devices has led to an exponential increase in the number of sensors used in our daily lives in recent years. With the rise and projected scale of the Internet of Things (IoT), many technological solutions are proposed for various smart applications such as smart street lighting, smart cities, and infrastructure monitoring [2]. Many early adopters of IoT used short-range radio technologies for their sensors that required multiple hops before reaching the Internet. However, recent advances have enabled a simpler technological solution called Low Power Wide Area Networks (LPWAN) that offer low energy communication over several kilometres, allowing sensors to send data to a central server over just a few hops.

This article was presented in part at The 2nd ACM/IEEE International Conference on Internet-of-Things Design and Implementation [1].
P. J. Marcelis, V. S. Rao, N. Kouvelas and R. V. Prasad are with the Embedded Software group of the Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, The Netherlands. E-mail: pjmarcelis@gmail.com, and {V.Rao, R.R.VenkateshaPrasad}@tudelft.nl.
Manuscript received XXXX; revised XXXX.

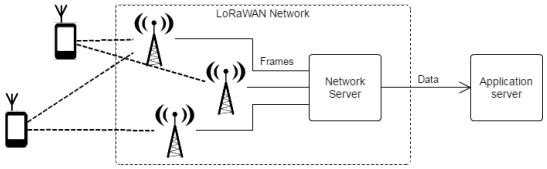


Fig. 1: Schematic overview of a LoRaWAN network.

LoRaWAN (Long Range Wide Area Network) is one of the several competing LPWAN technologies with, amongst others, SigFox, NB-IoT, and Weightless [3]. LoRaWAN has been the most successful of these technologies in providing an easily accessible LPWAN [4]. This protocol is being developed by the open LoRa Alliance [5]. In comparison with other LPWAN technologies, LoRaWAN claims to provide an inexpensive, secure and power-efficient communication method for applications with small end-devices that have to send small amounts of data at large time intervals. Therefore, it is important to study LoRaWAN and characterise it in practical deployment.

LoRaWAN is based on the LoRa physical layer, which describes a chirp spread spectrum modulation. This modulation technique is said to give LoRa a good resilience to interference, multipath and Doppler effects [6], [7]. The topology of a LoRaWAN deployment is shown in Figure 1. A LoRaWAN frame can be received by multiple gateways that forward the frame to a central LoRaWAN network server. The network server checks the device policy, decrypts the frame payload, removes duplicates, and forwards the data to the corresponding application server.

Since LoRaWAN communication takes place over long distances, frames can be lost due to propagation and other physical phenomena such as shadowing, reflection, and scattering due to urban clutter. Frame loss leads to a loss in data. Since an IoT application is often data-driven, the data loss must be minimal, so it is desired to recover the lost data. There are several constraints that need to be met for a LoRaWAN data recovery mechanism:

1) LoRaWAN targets to serve up to 15,000 devices per gateway. Automatic Repeat Request (ARQ) with acknowledgment (ACK) mechanism is an option in the LoRaWAN protocol [8], [9]. However, ACK requires downlink communication and also introduces communication overheads. With a large number of devices in a LoRaWAN network,

the transmission of ACKs for each packet seems improbable due to the duty-cycle limits (1% for gateways in Europe), and also that the gateways are typically half-duplex, which will lead to frame losses in the uplink communication [10]. Hence, ACKs are not a suitable method.
2) The solution must not need any change in the gateways since such a solution is expensive for existing deployments.
3) The solution must be able to deal with bursty frame losses, as we show in Section 5, that is due to the unreliable LoRaWAN channel.

Considering these constraints, we focus our research on a technique that should operate at the application layer such that the solution can work with already deployed LoRaWAN networks. Therefore, we aim to find the best method for increasing the data reception while trading off energy consumption. We identify the main research question as follows:

What application layer solution can provide data recovery from lost LoRaWAN frames with minimal additional energy consumption?

To address this question, we sub-divide it into the following questions:
1) How much frame loss can be expected in a LoRaWAN network?
2) Which coding method would guarantee sufficient data recovery for the given requirements?
3) What performance (in terms of data recovery ratio and latency) can be expected from implementation of this coding method?

We address these questions in this article. This extended article includes a more in-depth discussion on implementation, choice of the optimal degree, and new results with respect to bursty channels that were not part of the conference version of this paper. Specifically, our contributions through this work are as follows:
1) To study and characterise frame loss, we perform real-world measurements with a LoRaWAN network. We have collected data extensively (around 23,000 frames) over several days in stationary and mobile scenarios. Since the network was still in development, the LoRaWAN services were not yet open for the general public. This allowed us to do a first of its kind measurement of the LoRaWAN deployment in an *almost* collision free network. To the best of our knowledge, we are the first to perform such a large-scale measurement of a LoRaWAN network, which will be hard to do in the future.
2) We characterise both spatial (i.e., frame loss over distance) and temporal (i.e., burstiness of frame loss) properties of the channel using the datasets. With the collected data, we observe that there is a significant amount of frame loss that occurs as the end-device moves farther away from a gateway. We observe a loss of up to 53% when the end-device is around 6 km away from the nearest gateway. We find that the channel can be bursty even when the end-device is stationary.
3) To recover data from the lost frames, we propose a novel application layer coding technique called DaRe, based on convolutional and fountain codes. The need for such a scheme is further motivated by the ALOHA-type medium access technique employed in LoRaWAN, which will inevitably cause many collisions leading to frame loss. DaRe does not intend to recover the lost frames, but it enables the recovery of the data from lost frames using forward error correction on the application level. We use an algebraic framework to describe and simulate the coding technique. Since the coding method will be implemented on embedded devices with low computational capabilities, we design DaRe to be of low complexity.
4) We develop an implementation of DaRe for LoRaWAN. We evaluate this implementation by (i) emulating results for theoretical channels, (ii) emulating DaRe on the datasets collected during the network measurements, and (iii) performing frame and data loss measurements with a device running DaRe. DaRe can recover up to 99% of the data when up to 40% frame loss with a code rate of 1/2. With a code rate of 1/5, we can recover 99% of the data when up to 68% frames are lost. Furthermore, we show that the solution has minimal energy overhead compared to other low-complexity solutions.

The rest of the paper is organised as follows. In Section 2, existing research on LoRaWAN and erasure coding is discussed. In Section 3 a brief introduction on LoRaWAN is given. The method of data collection is explained in Section 4. An overview of the frame loss characterisation of LoRaWAN is provided in Section 5. In Section 6, DaRe and the algebraical framework is introduced. The performance of DaRe is assessed through numerical evaluation in Section 7. In Section 8, an implementation of DaRe is proposed and in Section 9, DaRe is evaluated. We conclude in Section 10.

2 Related Work

In this section we briefly discuss existing studies on LoRa, LoRaWAN and erasure coding to position our work.

2.1 LoRaWAN

Bor *et al.* [3] has done extensive research on the various IoT radios and they conclude that LoRa has longer communication ranges and other interesting features over other solutions, such as SigFox and Weightless. Aref *et al.* [11] performed free space range measurements with LoRa by testing different physical layer configurations. They concluded the Semtech SX127x LoRa radio family shows significant benefits for range, robustness and battery lifetime compared to competing technologies.

Centenaro *et al.* [4] tested the coverage range of LoRaWAN in an urban setting (in Padova, Italy). They performed stress tests by putting devices in elevators and other challenging locations, and LoRa passed all their tests. They found a nominal coverage range of 1.2 km for indoor environments. Petäjäjärvi *et al.* [12] looked into the coverage range of LoRaWAN in indoor environments as well. In their test setup, a minimum of 96% frames was received indoors for distances up to 420 m. They also looked into the range of LoRaWAN outdoors [13] and observed a maximum communication range of 15 km on ground and close to 30 km on water.

Mikhaylov *et al.* [14] researched the capacity and scalability of LoRaWAN, analysing the limits of device and gateway throughput, giving the first insights on potential

LoRaWAN network performance. They concluded that a LoRaWAN network shows high coverage and satisfactory scalability under low uplink traffic. The drawbacks were the low reliability and the potentially poor performance for an increasing load.

Augustin *et al.* [15] did a comprehensive study on the properties of LoRa and LoRaWAN, finding it suitable for low-power, low-throughput, and long-range networks. In simulations on collision rates, they stated that acknowledged messages significantly reduce the successful throughput in a LoRaWAN network and should, therefore, be avoided.

In the existing studies, LoRaWAN is considered a good solution for an LPWAN in general. However, the reduced communication reliability for an increasing number of devices is identified as a challenge. The conducted measurements in the literature involve only setups with a single device, or the spatial properties of a single gateway [16], [4], [13], [17]. The primary goal of Petajajarvi *et al.*'s work [13] was to find the maximum transmission range with respect to one gateway. They also perform a frame loss characterisation but do not study the effect of mobility or the temporal properties of the channel. The work of Rathod *et al.* [17] comprises of preliminary experiments inside campus for short ranges. Again, they do not consider mobility or temporal properties of the channel. On the behaviour of LoRaWAN networks, only simulations have been done [15], [14]. In this article, measurements are presented on a LoRaWAN network with multiple gateways. Also, we conduct the first analysis of LoRaWAN communication with frame loss of bursty nature.

2.2 Erasure Coding

Erasure channels characterise and model communication channels, describing how transmitted messages are either received, or erased (lost). A specific type of communication error is the erasure. While a simple error is a misinterpreted piece of information, erasure is a lost piece of information. A channel in which parts of information are erased is called an erasure channel. An erasure channel is considered as binary-based or packet-based, depending on what is erased (bits or frames). Since the proper protocol data unit name for LoRaWAN is a frame, LoRaWAN is a frame erasure channel. In error channels, messages are altered but the frames are still received unlike in erasure channels. For the sake of readability, we shall refer to erasure as frame loss and erasure channel as frame loss channel henceforth.

In communication theory, majorly, two methods exist to deal with frame losses: Automatic Repeat reQuest (ARQ) and Forward Error Correction (FEC) [18]. The former employs error detection schemes in communication but relies on frame retransmission to correct errors, which is inefficient because of data overhead and more collisions. In FEC, redundant data is transmitted that helps in detecting and correcting a limited number of errors. An FEC method can be used to add robustness in LoRaWAN communication.

Erasure/frame loss coding deals with frame loss channels. We also find frame loss coding in distributed data storage, where information is stored redundantly over different disks or servers. This allows the loss of storage media while preventing data loss. In applications communicating over packet-switched networks, like media streaming [19], frame loss coding is employed to handle packet loss without data loss.

The earliest erasure codes are Reed-Solomon codes [20], introduced in 1960. Reed-Solomon (RS) codes are, for instance, found on CD-ROMs, to withstand physical damage to the disk. RS-codes, however, requires complex computations and are resource expensive [21]. Thus, they are not suitable for LoRaWAN.

In 1962, Gallagher developed the concept of Low-density parity-check codes (LDPC codes) [22]. With LDPC codes, data blocks are supplemented with parity bits before transmission to enable error/frame loss detection and correction. LDPC codes are, for instance, used in the current DVB-S2 standard [23]. RS-codes and LDPC codes provide error/frame loss detection and correction. In LoRaWAN, frame loss detection is done separately from correction.

2.2.1 Fountain Codes

In Fountain codes, error detection is presumed given. Thus, the coding scheme is designed completely for erasure correction. In fountain codes, codewords are calculated as linear combinations of data fragments [24]. The principle of a fountain code is that any random combination of sufficient codewords can be used to decode the information [21].

Fountain codes were introduced by Luby [25]. Examples of fountain codes are LT-codes [26], Raptor codes [27] and Online codes [28]. In fountain codes, the idea of producing an infinite number of code words, instead of a fixed number, was introduced (rateless codes). Fountain codes are block codes, like LDPC codes.

The mathematic principles of fountain codes can be used to solve LoRaWAN data loss. The content of frames can be viewed as data fragments. Linear combinations of these data fragments can be transmitted as well to provide redundant information. The redundant information needs to be spread over multiple LoRaWAN frames.

2.2.2 Convolutional Codes

An alternative to block codes is the convolutional codes, which encode with a sliding window. This means the dataset used to calculate codewords changes over time, while for a block code the dataset is constant. An example of a convolutional code is the class of Turbo codes [29]. Turbo codes are iterative, with messages being sent along with parity bits computed in a recursive manner.

In our application, we have periodically generated data. Using a block code (e.g., fountain codes), we would have to wait for some data to be generated before coding could take place. On the other hand, in convolutional codes, coding can already be done from the first piece of generated data.

The principle of a sliding window can be applied on fountain codes, making a convolutional-fountain code. There are already some works on combining fountain codes and convolutional codes [30], [31]. Also, there has been research done on applying fountain codes in a windowed manner [32]. These existing studies do not apply fountain codes on a sliding window in a convolutional manner. A convolutional fountain code would provide the suitable coding method for our problem statement. Sandell

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3016654, IEEE Transactions on Mobile Computing

et al. [33] also studied the combination of convolutional-fountain codes for LoRaWAN, in which they analysed our work on DaRe [1] focusing on the effect of the code rate on the system performance.

2.2.3 Forward Error Correction in other IoT Technologies

IoT communication technologies, involving Bluetooth and IEEE 802.15.4 devices, utilize FEC in combination with interleaving for data recovery. In 802.15.4 networks, the end-devices employ FEC mainly to tackle the interference of powerful Wi-Fi signals [34], [35], [36]. Furthermore, they employ FEC to recover corrupted bits/data from erroneously received frames. FEC and ARQ mechanisms are also used by Bluetooth separately or in hybrid schemes [37], [38], again to recover corrupted bits from frames received with errors.

These works do not target to recover data but frames. There are two main differences between the works employed in 802.15.4 and Bluetooth networks and our consideration in LoRaWAN: (a) The channels are bi-directional and ARQ with ACKs are commonly used; and (b) FEC is mostly used to avoid ARQ through recovery of frames and be energy-efficient. However, the lack of a feedback channel in LoRaWAN, due to the regulations, allows us with one choice to increase reliability. Furthermore, as entire frames are lost in the channel (Section 5.1), adding more redundant bits in a frame will not help, which is already being done by LoRaWAN specifications. Hence, we focus on application layer coding for LoRaWAN.

3 An Overview of LoRaWAN

In this section, we briefly introduce LoRaWAN. A LoRaWAN network has a star of stars topology: the data transmitted by the end-devices may be received by one or more gateways, which in turn connect directly to a central network server. A schematic of this network topology is shown in Figure 1. The network server removes duplicate messages and forwards them to an application server.

LoRaWAN operates in the license-free industrial, scientific, and medical (ISM) band. The 915 MHz and 868 MHz bands are used in the USA and in Europe, respectively. The band in Europe has certain regulations [39]. We refer the reader to [4] for details on listen-before-talk option and the regulations in the rest of the world. In Europe, a maximum transmit power of 14 dBm is allowed on the end-devices when LoRa modulation is used, and the maximum allowed transmit duty cycle is 1% (up to 10% on one channel), which applies to the gateways as well.

Devices communicating over LoRaWAN have three operating classes –A, B, and C– with increasing order of downlink communication (i.e., from a gateway to a device). Class A is the default class of operation. A device in this class adopts an ALOHA-like scheme to send a LoRaWAN frame. After a Class A device has transmitted a frame, it goes into 'listening' mode for two time slots. Only in these time slots can the device receive data. Class B extends Class A by adding extra receiving-slots at preset times. The 'extra listening' duration is specified by the gateway using beacon frames that are transmitted to the end-device. A Class C device will always be in receiving mode, unless it is transmitting. Class A is the most preferred operating class for battery-powered devices since it has the lowest energy consumption.

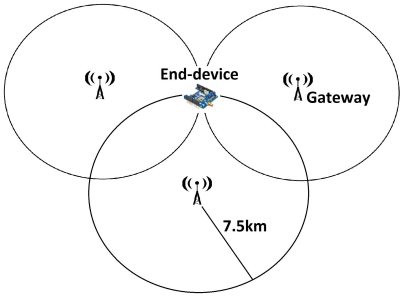


Fig. 2: Gateways are deployed such that the maximum distance for an end-device from its closest gateway is 7.5 km.

A LoRaWAN frame is transmitted with a certain spreading factor (SF). The spreading factor ranges from SF7 to SF12, expressing the number of chirps used to encode a symbol (i.e., 2^{SF} chips $=$ 1 symbol). A higher spreading factor has a longer transmission time and lower data rates but will give a longer range. LoRaWAN supports the automatic setting of the spreading factor with Adaptive Data Rate (ADR). The network server monitors the quality of the uplink communication and informs the device on the best spreading factor to use.

The LoRa physical layer employs a soft hamming block error correcting code with a code rate 4/5 to reduce bit errors. However, this does not eliminate frame loss completely. The reception of a LoRaWAN frame relies primarily on detecting the preamble of a frame [40]. If the preamble is not detected by the receiver, the complete frame is not received. This makes the LoRaWAN communication channel a frame loss/erasure channel. Frames are either completely received, or not received at all.

LoRaWAN packets also offer the option of a header with a 16-bit cyclic redundancy check (CRC) for error detection. Message-integrity is preserved by means of a 32-bit message integrity code (MIC). This MIC also guarantees message authenticity, since it is a cipher-based message authentication code (CMAC), calculated with a device-specific key.

4 Setup and Data Collection Scenarios

In this section, we describe our data collection setup, scenarios, method and the datasets collected for the analysis of the LoRaWAN communication channel.

4.1 Measurement Setup

4.1.1 Network

The LoRaWAN network used for our research was in development, so there were only a handful of devices transmitting data on the network. Therefore, we consider frame loss as a result of collisions to be negligible. Furthermore, the configuration of the network was still being improved and the gateways were positioned only for (close to) line-of-sight coverage.

While the measurement devices adhered to the duty cycle limits, they were exempted from any usage limits on

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3016654, IEEE Transactions on Mobile Computing

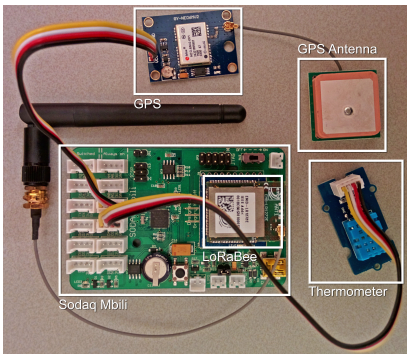


Fig. 3: The data collection device. A Sodaq Mbili Rev. 4 [41] with EMB-LR1272E LoRaBee module, DHT11 thermometer and GY-NEO6MV2 GPS module.

the number of messages. We use Thingpark Wireless Logger[1] to log the LoRaWAN frames received by the network server. Wireless Logger stores payload and metadata of each received frame. As explained before, multiple gateways can receive a single frame. Unfortunately only up to three receiving gateways per frame are shown in the logger. That is, only the three gateways that received the LoRaWAN frame with the strongest received signal strength indicator (RSSI) can be identified in the logs. This complicates a per gateway analysis of frame reception.

All the gateways in the LoRaWAN network are situated at an average height of 27 m. The antennas have a gain of 11 dBi. The gateways are positioned on an average of 8 km apart. Furthermore, the maximum distance between an end-device and its closest gateway is 7.5 km as the gateways are carefully positioned as shown in Figure 2.

4.1.2 End-Device

The end-device used for data acquisition is a Sodaq Mbili Rev. 4 [41], shown in Figure 3. The antenna has a gain of 2 dBi[2]. The device operates in Class A.

To characterise the LoRaWAN communication channel spatio-temporally, measurements were done at multiple locations. To perform the analysis after data collection, we need the location from which each frame was transmitted. A first version of the device did not include a GPS module, so the location of every transmitted LoRaWAN frame was matched to a GPS track created with the mobile application Strava[3] using timestamps. The final version of the device included a GPS module, so the coordinates were sent in the frame payload itself.

The LoRaBee module was a separate component and was connected to the Sodaq Mbili with a serial interface. The serial interface was used to transmit data, retrieve the received data, and to set MAC parameters. The sub-band and bandwidth were set to comply with ETSI standards [42]. The bandwidth was 125 kHz for all transmissions. Furthermore, we set the transmission power to the maximum allowed value of +14 dBm.

1. http://www.thingpark.com/
2. A gain of 2 dBi antenna is within the European regulations, which states that the maximum effective radiated power should be 14 dBm [42], [43]
3. https://www.strava.com/

The SF is controlled by the ADR mechanism, since a fixed value is not supported by the device. However, the largest part of the frames (∼95%) was transmitted using SF12. Only these frames have been used for data analysis.

The SF and the coordinates of the unsuccessful frames were interpolated on a straight line between the neighbouring received frames. The SF for the missing frames was taken to be the lowest SF of the neighbouring received frames.

4.2 Data Collection Scenarios

We identify two scenarios for our data collection: data from stationary locations and data when the end-device is moving (mobile data). All the data was collected with the end-devices placed next to a window or outdoors.

4.2.1 Stationary data

The stationary datasets were generated with the end-devices transmitting mostly at every 10 minutes or 15 minutes. A small part of the dataset contains frames transmitting at 15 s interval. During measurements, the devices were in a constant position and orientation. The stationary dataset (∼18,000 frames) was collected at different houses (locations) in the city of Delft, The Netherlands, which forms a typical urban scenario. While the gateways were positioned to be for LoS coverage, the urban clutter cannot guarantee LoS transmissions at all locations. The set of rich data captures the complex propagation environment in urban scenarios, including non-line of sight transmissions.

4.2.2 Mobile data

The mobile datasets were collected using the device shown in Figure 3. During data collection, the device was battery powered. Measures were taken to ensure that the battery was sufficiently charged and operational during the entire duration of data collection. For generating mobile data we chose two methods to make it mobile: by bicycle (bike) and by car.

A bike has been ridden with an average speed of 22 km/h for approximately 300 km in total. The farthest distance to the closest gateway is 7.5 km, and the average distance to the closest gateway is 3.2 km. The end-device was also taken in a car for approximately 350 km in total. The average speed was around 80 km/h (mainly on highways). The selected terrain was flat. Among the 5,000 LoRaWAN frames contained in the mobile data, approximately 80% were transmitted from rural areas and the rest were sent from an urban area which included non-line of sight transmissions. The datasets were generated with the end-devices transmitting every 15 s.

5 Frame Loss Characterisation

Before we present our coding scheme we first provide an overview of the analysis of the datasets in order to characterise the frame losses. A more detailed analysis of the channel model and the frame losses can be found in [1]. Note that the characterisation presented here also holds for class B and C end-devices for the uplink channel.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3016654, IEEE Transactions on Mobile Computing

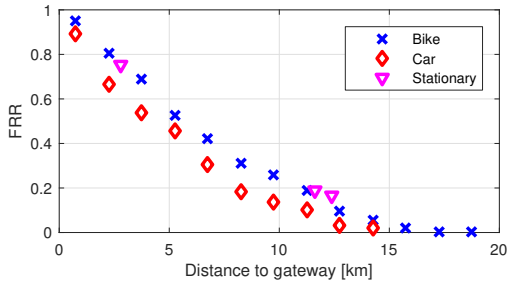


Fig. 4: Frame Reception Ratio (FRR) as function of the distance to a gateway for bicycle, car and stationary datasets.

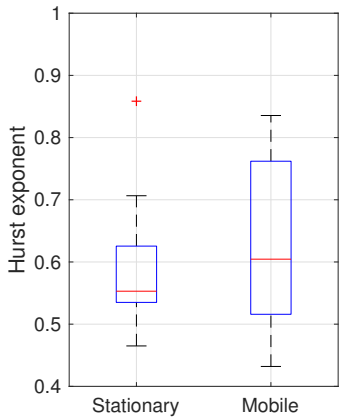


Fig. 5: Box plot quantifying the burstiness in the two different dataset types using the Hurst exponent.

5.1 Frame Loss over Distance

We use the bike dataset to determine the parameters of the path-loss channel model with shadowing [44] between the sender (end-device) and the receiver (gateway), since the dataset provides measurement points at different distances. The Doppler effect is negligible because of the relatively low speed. Since the bike data is collected from all different routes, we can negate the effect of permanent shadowing and other effects due to surroundings.

Using the analyses of the dataset as in [1], we estimate the path-loss exponent to be 2.71, and the shadowing follows the log-normal distribution with a mean of 0.56 and standard deviation of 7.11. With these parameters and the receiver sensitivity of the gateway (-137 dBm), we can estimate the cell outage probability [44]. The outage probability for the farthest distance of 7.5 km is found to be 0.004, indicating that the coverage of the current deployment is sufficient. While the frame loss characteristics would be typical as reported here, the path-loss exponent and the shadowing distribution may be affected by several factors including gateway location, gateway height, antenna gains, and terrain. A method to determine optimal deployment using satellite imagery has been proposed in [45].

To characterise the frame loss or erasure of frames, due to the channel effects, we analyse the data from all our datasets. We first consider frame loss as a function of distance between the end-device and the gateway. To account for location estimation inaccuracies we consider bins of 1.5 km in which we calculate the average frame reception ratio (FRR). Figure 4 shows the FRR with respect to the distance of the end-device from the gateway. It is evident from the figure that more frames are lost as the distance to the gateway increases. While the outage probability is quite low at 7.5 km, the frame loss is significant at that distance: almost 70% of the frames is lost around that distance. This seems counter-intuitive as the coverage was found to good theoretically. This is due to the fact that the theoretical calculations were based on a simple path-loss model and not considering the complex propagation environment. We can observe another interesting aspect in this figure because the datasets are collected at different moving speeds. The dynamics of the channel is pronounced when the end-devices are mobile due to varying channel fading and multipath fading, which is the main reason for the increased losses as the end-devices move at higher speeds, especially for higher SFs (due to longer airtimes).

Finding #1: Frame loss is quite significant in LoRaWAN despite an almost collision-free channel.

Finding #2: Frame losses are higher due to the dynamics of the channel when the end-devices move at high speeds.

5.2 Burstiness

We analyse the frame loss pattern over time in LoRaWAN. While some datasets exhibited uniformly random frame losses, other datasets showed more consecutive erasures than to the expectation for an independent and identically distributed (IID) frame loss probability. This prompted us to look at the burstiness of the frame losses. Burstiness is a temporal property of the channel. The channel shifts between poor and good states with a correlation between the frame delivery events. This results in frame losses being closer to each other for a bursty data stream compared to a non-bursty data stream. While burstiness in wireless links has been studied extensively, a metric to express burstiness is not standardised. We take a simple and effective measure to quantify the burstiness, using the *Hurst exponent*, or the self-similarity parameter, H [46].

This exponent is used to reveal self-similar streams of data, i.e., data patterns over smaller intervals which are also present over longer intervals. Considering a binary time-series, where a 1 corresponds to a received frame, and a 0 to a lost frame, data sets with burstiness will show lower self-similarity than data sets without burstiness, for which $H \sim 0.5$. The burstiness increases with any increase in the Hurst exponent and if $H > 0.6$, a dataset is characterised as bursty. In Figure 5, a box plot of the calculated values for the Hurst exponent for the different type of datasets (stationary and mobile) is shown.

We observe that the mobile data sets show more burstiness on average ($\overline{H_{bike}} = 0.6$) and higher deviation than the stationary data sets ($\overline{H_{stat}} = 0.56$). Burstiness in the mobile dataset is due to mobility, while burstiness in stationary datasets is the result of channel effects.

Finding #3: LoRaWAN channel is also prone to lose frames in bursts due to channel effects and mobility.

For the design of the data recovery method it is important to know if burstiness occurs. When burstiness is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3016654, IEEE Transactions on Mobile Computing

present, the data recovery method should spread redundant information more over time.

5.3 Modelling the Frame Loss Channel

Around half of our datasets show uniformly distributed frame loss. So the frame loss probability of a frame can be considered to be IID. A valid channel model for these datasets is a Bernoulli channel model with frame loss probability p_e. We define this as the default LoRaWAN communication channel.

For the datasets that show burstiness we use the Gilbert Elliot model [47], [48], [49]. The Gilbert Elliot model is based on a Markov chain with two states: a good state where there is no frame loss and a bad state in which a frame is lost with probability p_{loss}. The probability to transit from the good state to the bad state is expressed with p_{GB}, and the probability of transition from the bad state to the good state is expressed with p_{BG}. The channel model has an average frame loss probability p_e that can be calculated with $p_e = p_{loss}/(1 + \frac{p_{BG}}{p_{GB}})$.

The parameters of the model used to simulate the behaviour of this dataset can be determined empirically. For example, the empirically calculated Gilbert Elliot model parameters that describe the burstiness of the car dataset are $(p_{GB}, p_{BG}, p_{loss}) = (0.25, 0.21, 0.85)$.

6 DaRe: Data Recovery in LoRaWAN

DaRe is an application layer coding technique, which is a combination of both convolutional and fountain codes. To the best of our knowledge (as mentioned in Sec. 2), DaRe is the first such coding technique. DaRe can be applied to any application transmitting over a lossy transmission medium, however, with some fine-tuning. DaRe has been tuned for the unique properties of LoRaWAN. The main constraints are due to the packet size limitations imposed by the SFs (specifically for SF12), and the regulations on duty cycle.

Most applications of LoRaWAN use the class A mode in which an end-device transmits its sensor data periodically but infrequently using an ALOHA-like protocol. We refer to the sensor data that needs to be transmitted as a *data unit*. This data is at the heart of the IoT applications. Since frame loss leads to data loss, frame loss must be minimised.

In the previous section, we have demonstrated that the frames transmitted over LoRaWAN can experience significant frame loss. As mentioned in Sec. 1, frame ARQ mechanisms using acknowledgement are possible over LoRaWAN, but cannot be employed for all frames. Gateways can only acknowledge a limited number of frames as the gateways must obey the ISM-band duty cycle limit. Even if a workaround is used by employing other channels, retransmissions are still not scalable since the frames may collide in the uplink with the increasing number of devices. Furthermore, acknowledgement and ARQ mechanisms increase the energy consumption for the end-devices, which is undesirable.

This necessitates a solution to minimise data loss due to frame loss without employing acknowledgements. As mentioned in Sec. 1, there are several requirements for such a solution: (a) No changes to the LoRaWAN specifications or network should be needed in order to operate with an already deployed infrastructure; (b) increase in transmissions must be minimal so as to be scalable and have minimal overhead in terms of energy; and (c) the solution must be of low complexity at the transmitting side, since the end-device is an embedded device with limited computational and energy capabilities.

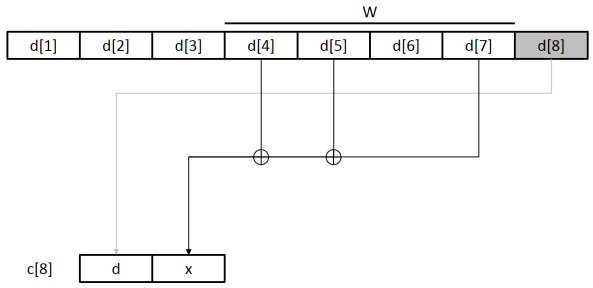


Fig. 6: A schematic explanation of DaRe. In this example, R=1/2, W=4, Δ=0.75. The code words for the frame at time instance $t = 8$ are calculated by concatenating the data units from $t = 8$ and a parity check of previous data units from $t = 4, 5, 7$.

To this end, we propose a coding technique, called DaRe (Data Recovery), that borrows from both fountain and convolutional coding techniques, and operates at the application layer. For data recovery, an existing set of data units should be extended with redundant information, such that the original set of data units can be recovered even if only a subset of the transmitted data units is received. LoRaWAN operates on a packet frame loss channel, i.e., frames are either received or completely lost. Thus, DaRe must spread the redundant information from the data in one frame across other frames. Thus, a lost frame can be recovered using the redundant information from other frames.

The redundant information included in a frame is a parity check of randomly selected previous data units. A parity check is a vector of parity bits for each bit position in the data units. Traditional fountain codes perform the coding over a data block. But since we want the redundant information to be calculated in a convolutional manner, we use a sliding window approach with a finite window.

Since we intend to keep the complexity low for embedded devices, we work only with Galois Field 2 (GF(2)). This implies that the multiplications and additions are bitwise 'AND' and 'XOR' operations respectively. A schematic explanation of determining the frame payload using DaRe is shown in Figure 6. The coding parameters are explained more in detail in the following section.

6.1 Coding Parameters

There are three parameters in the context of DaRe: code rate (R), window size (W) and degree (Δ). We define them and discuss their influence on the coding scheme in this section.

6.1.1 *Code rate*

The code rate (R) is the ratio between the size of original data and the size of the data actually transmitted. It expresses the amount of redundant information added in transmission, and is calculated with the following equation:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3016654, IEEE Transactions on Mobile Computing

$$R = \frac{\text{number of data bytes}}{\text{number of transmitted data bytes}}. \quad (1)$$

If we have data units of size y bytes, the size of the frame content after adding the redundant information will be y/R. A lower valued code rate is expected to provide higher data recovery percentages.

If the size of the data units, y, is one byte, the code rate can only be of the form $1/(y+x)$, where x is the number of redundant data units appended to the frame. This is because the smallest unit that can be easily processed on an embedded device is one byte. If y is more than one byte, data units can be split into k fragments if y is divisible by k. Then the code rate can be of the form $k/(k+x)$. In this case the calculation of the parity checks is done at the data fragment level instead of data unit level. This allows for a higher number of code rate values, including code rates higher than $1/2$, making the transmitted frames smaller with some redundancy.

Since the code rate determines the size of a transmitted frame, the code rate is limited by the maximum possible size of the frame. In LoRaWAN, the maximum allowed frame content is 51 bytes at SF12 and 242 bytes at SF7. The other SFs have a size between these two extreme values.

6.1.2 Window Size

The window size (W) expresses the number of previous data units to consider for calculating the redundant information. The window size is limited by the available memory on the end-device. A larger window size will spread redundancy for a single data unit over more frames, increasing the recovery probability of this data unit. A larger window size will also increase the maximum length of consecutive frame losses that can be recovered. However, increasing the window size will also increase the recovery delay of the data unit. Theoretically, there is no limitation on the maximum window size. Considering an infinite window size and infinite number of messages received, all the lost data can be recovered as sufficient number of parity bits for all frames would have been received.

6.1.3 Degree

The degree (Δ) expresses the relative number of previous data units from the selected window to include in the parity check. For example, if W=10 and Δ=0.5, there will be 5 data units included in a parity check.

Although specific fountain code implementations propose a special degree distribution, all these implementations are patented [24], [21]. Thus, for DaRe we choose the degree to be constant over different parity checks.

The exact combination of data units in a parity check is picked randomly for each parity check. This is done to eliminate the influence of certain patterns in the selection. This leads to an average result for all different sequences of exact combinations.

We can compute an optimal value for the degree Δ for a given window size W. The optimal value implies there exists a value of Δ that offers the best recovery results for a given window size. By using this optimal value of Δ in DaRe, we only need to input two parameters for coding: the window size W and the code rate R (see Section 8.4 on how we choose parameters for implementing DaRe).

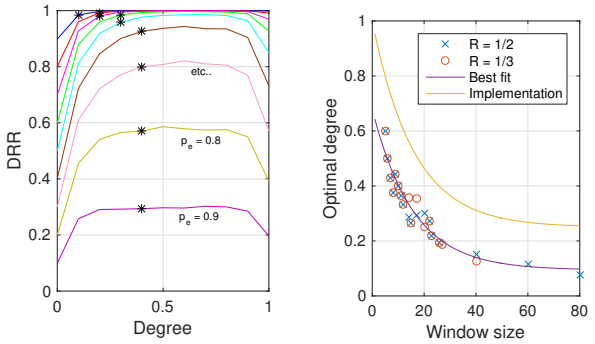


Fig. 7: Determining optimal degree value Δ_{optimal}. (a) DRR for R=1/2 and W=10 for different frame loss probabilities when varying Δ. The black crosses mark the lowest values for Δ that give the maximal DRR for different frame loss probabilities. The largest of these values is Δ_{optimal}=0.4. (b) Δ_{optimal} for different values for R and W. There is an exponential relation between Δ_{optimal} and W. A relation with rounded-off fixed point coefficients is chosen as implementation.

The degree has a major influence on the performance of the coding scheme. In Figure 7(a), the data recovery ratio (DRR) is plotted for a fixed code rate and window size while varying the degree, over channels with different IID frame loss probabilities. The results are from numerical evaluation as described in Sec. 7. An extreme degree value gives a lower DRR. With a degree of 0, there is no redundant information so the data recovery ratio is equal to DRR $= 1 - p_e$. For degree values close to 1, the recovery ratio decreases as well. For each value of p_e there is a range for the degree value that gives the maximum possible DRR. The lowest value for Δ that gives the maximum possible DRR for a specific p_e is $\Delta_{\text{optimal}}(p_e)$ and is plotted as black crosses in Figure 7(a). The value that gives the maximum possible DRR for all p_e is Δ_{optimal}. For R=1/2 and W=10 the value for Δ_{optimal}=0.4.

In Figure 7(b), the value for Δ_{optimal} is plotted for different W and R. We observe that R does not influence Δ_{optimal}, so Δ_{optimal} is only subject to W. We can also observe from the figure that the optimal degree value Δ_{optimal} and the window size W have an exponential relation plotted as "Best fit".

To reduce the number of floating point computations for the embedded device, we round-off the coefficients of the relation between W and Δ_{optimal} to the closest fixed point binary representation. Although the coefficients are rounded-off, the value for Δ still lies in the range that gives maximal recovery rates for all frame loss probabilities. The fixed point relationship is given by:

$$\Delta_{\text{optimal}}(W) = \left(\frac{1}{2} + \frac{1}{2^2}\right) \exp\left(-\frac{1}{2^4}W\right) + \frac{1}{2^2}, \quad (2)$$

and is plotted in Figure 7(b) as 'Implementation'.

The optimal degree value for a given window size is a maximisation problem in which the DRR must be maximised subject to a given window size and frame loss

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3016654, IEEE Transactions on Mobile Computing

TABLE 1: Glossary of symbols

Symbol	Description
$d[t]$	Data unit at a given time instance t of size k data fragments
$D[t]$	$D[t] = \{d[\tau] \mid \max(t-W-1,0) \leq \tau \leq t\}$
$c[t]$	Coded data at a given time instance.
$\mathbb{G}_t$	Generator matrix of time instance t describing relation between $D[t]$ and $C[t]$
D	$D = D[\infty]$
C	$C = \{c[\tau] \mid 0 \leq \tau \leq \infty\}$
$\mathbb{H}$	Matrix describing relation between D and C
C^*	The received coded data. $C^* = C \cup *$
$\mathbb{H}^*$	Relation matrix describing relations between C^* and D

probability. Multiple solutions may exist, as shown in Figure 7, and minimum of these solutions must be chosen as the degree. While this is easy to compute, the optimal degree must be computed each time a different frame loss probability is chosen while the window size remains the same.

Alternatively, an optimal degree for a given window size can be computed using the minimax framework. The problem is to minimise the maximum degree over a range of frame loss probabilities for a given window size. Computing one value as the optimal degree for a window size is preferred over computing for each frame loss probability, which gives negligible gains. We shall address this in our future work.

6.2 Mathematical Framework

The framework presented here is a general framework to perform application level frame loss coding and is applicable for Galois Fields higher than two as well. A glossary of symbols used in this section is provided in Table 1.

Let $d[t]$ represent the data units generated at time instant t. All the generated data units are of equal size and are divisible in k data fragments. By concatenating data units, we get the dataset $D[t]$ that contains all generated data units previous to time instance t. $D[t]$ has maximum length $W+1$. Only the W previous data units need to be stored to calculate the redundant information. At time instance t data unit $d[t]$ is transmitted concatenated with redundant information. This set is called $c[t]$. The redundant information is a parity check, so it can be expressed as a linear combination of previous data units with the following equation:

$$c[t] = D[t]\mathbb{G}_t, \tag{3}$$

where $\mathbb{G}$ is the generator matrix of size $k \times n$, and $n = k/R$, $\forall t$.

Let the set C be a concatenation of all values of $c[t]$, and the set D be the set of all data units. We can state that,

$$C = D\mathbb{H}, \tag{4}$$

where,

$$\mathbb{H} = \begin{bmatrix} | & & & & \\ \mathbb{G}_1 & | & & & \\ | & \mathbb{G}_2 & | & & \\ & | & \mathbb{G}_3 & | & \\ & & | & \mathbb{G}_4 & \\ & & & | & \ddots \end{bmatrix} \tag{5}$$

Through the frame loss channel some data will be lost. The decoder will receive only a subset of C. This gives a set $C^* \subseteq C$ of received coded data. For each $c[t] \in C^*$ the corresponding generator matrix $\mathbb{G}_t$ should be retrievable. All these generator matrices can be concatenated into one matrix $\mathbb{H}^*$ containing the relations between the data units at each time instance and the received code words. If matrix $\mathbb{H}^*$ is invertible, all the data units can be recovered using the following expression,

$$D = C^*(\mathbb{H}^*)^{-1}. \tag{6}$$

If $\mathbb{H}^*$ is not invertible, not all data can be recovered. However, the rank of matrix $\mathbb{H}^*$ expresses the amount of unique data in the matrix and is a metric for the amount of data that can theoretically be recovered by the decoder.

6.3 DaRe in Mathematical Framework

The mathematical framework can be used to simulate DaRe by describing a specific form of the generator matrix $\mathbb{G}_t$. Since the coding scheme is a systematic code, the first columns of the generator matrix should be an identity matrix in order to include the fragments of the data unit $d[t]$. The other code words are parity checks of previous data units.

This generator matrix for DaRe is of the form,

$$\mathbb{G}_{t,\text{DaRe}} = \begin{bmatrix} \mathbb{I} & 0 \\ 0 & P \end{bmatrix}, \tag{7}$$

where $\mathbb{I}$ is the identity matrix of size $k \times k$. P is the parity check matrix in which each of its $n-k$ columns contain parity check lines computed from randomly selected $k\Delta$ data fragments from a window of kW data fragments. We identify this metric as the data recovery ratio (DRR), defined as following:

$$\text{DRR} = \frac{\text{Number of data units recovered}}{\text{Number of data units transmitted}}. \tag{8}$$

6.4 Benchmark: Conventional Repetition Coding and LT Codes

A simple form of frame redundancy is repetition, i.e., to append previous data units to a frame. This method provides some redundancy and allows for recovery, but gives a lot of overhead. This method can also be expressed in the proposed mathematical framework. The generator matrix $\mathbb{G}_t$ would be time invariant and equal to an identity matrix $\mathbb{I}$ of size $n \times n$. We use this coding method as a benchmark for DaRe as it provides a performance reference. Additionally, we also compare the performance with Luby Transform (LT) codes [25], which are widely used fountain codes. The LT codes have been adapted to operate with finite windows for fair comparisons.

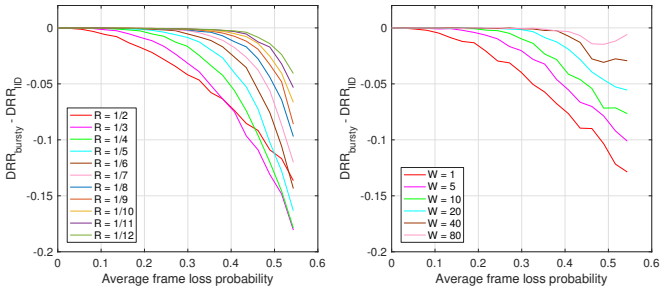
7 Numerical Results

We perform numerical evaluation with the mathematical framework in MATLAB to compare DaRe with repetition coding algorithms, used with LoRaWAN.

We observe in Figure 8(a) that burstiness impacts the coding method used in the benchmark quite significantly, with an additional loss of up to 18% for higher values for the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3016654, IEEE Transactions on Mobile Computing

code rate. Compared to DaRe, shown in Figure 8(b), DaRe offers much better resilience. For a window size of W=80, the maximum performance reduction is 1.4%. Therefore, we can conclude that DaRe can handle both bursty and non-bursty frame losses equally well.



(a) (b)

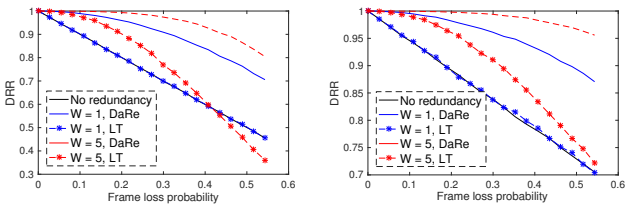
Fig. 8: Difference in DRR over a Gilbert Elliot channel model compared to an IID channel, with the Gilbert Elliot model parameters $(p_{GB}, p_{BG}) = (0.25, 0.21)$ varying p_{loss} and plotted for p_e. (a) For the benchmark. (b) Using DaRe with different values for W and R=1/2.

In order to compare, we chose Luby Transform Codes [25], which is one of the most popular fountain codes, and modified to operate over a given window size (W = 1 and 5). The method of implementation of this modified LT codes is as follows: The LoRaWAN nodes send sensor data at a pre-determined periodicity. This data is also stored in a buffer of size W. After the window is full, LT encoding is performed. For the next W packets, a block of encoded data is sent along with the sensor data. The blocks are collected at the receiver and then are passed on to the decoder. If the decoding is successful, then all the packets from the previous window can be recovered.

The results of evaluation for regular and Gilbert channels are shown in Figure 9(a) and Figure 9(b). We observe that DaRe outperforms the convolutional LT codes significantly.

8 Implementation

In this section, we present implementation details for a DaRe encoder and decoder on an end-device and application server respectively. Furthermore, we present a method that helps to adapt the parameters of DaRe on a real LoRaWAN network.



(a) (b)

Fig. 9: Comparison of DaRe with LT codes. (a) For regular channel. (b) For Gilbert channel.

To have DaRe working in real-life there are two challenges: (a) The decoder must be able to construct the generator matrix similar to the one used for encoding in order to decode the redundant information, and (b) An algorithm is needed to recover previously lost data units from successfully received frames. We describe the aspects to solve these challenges.

8.1 Generator Matrix at Decoder

In DaRe coding, the generator matrix is constructed as described in Equation (7). The generator matrices are randomly generated by the encoder at runtime. The generator matrices can be known to the decoder in two ways: (a) the generator matrices are sent along with the coded data in the frame, or (b) the generator matrix is generated with a pseudo-random function. The seed for the pseudo-random function should be sent in the frame. Since LoRaWAN frames have a limited payload size and we want to minimise additional transmission, solution (b) is preferred.

We need to create a pseudo-random function that generates the DaRe generator matrices. To be precise, a pseudo-random function is needed to generate the generator vectors, since these are the random part of the generator matrices.

The basis of a pseudo-random function is a pseudo-random number generator (PRNG). A good implementation of a PRNG for an embedded device is a linear-feedback shift register (LFSR) [50]. The output of a LFSR is pseudo-random sequence of bits that are calculated as a linear function of the internal state of the LFSR. Multiple output bits can be concatenated to generate a random number. Variables for a LFSR are the initialisation state, the output function and the number of state iterations. The initialisation state describes the internal state of the LFSR after zero iterations. The output function describes the linear function to calculate the output, which is also the input for the next iteration. The period of a LFSR is maximum length of an output sequence that is not repeated. The period is determined by the output function. Certain output functions give a maximum-length period of $2^n - 1$, where n is the number of shift registers in the LFSR.

Every LoRaWAN frame has a unique, incremental identifier, a frame counter, which is 4 bytes in size (32 bit). This value can be used as seed for the LFSR, since it is unique for each frame; and since there can be more than one parity check in one frame, the parity check index should be used as a seed as well.

Since window sizes larger than W=40 did not offer better results, the period of the LFSR does not have to be much larger than 40. By choosing an 8-bit LFSR with maximum period, we have a period of $2^8 - 1 = 255$. An 8-bit LFSR is easily implementable in present-day embedded platforms. The autocorrelation of an output sequence of the LFSR is given in Figure 10(a) for different window sizes W. The period of 255 can be seen in the figure.

The pseudo-random numbers that are outputted by the LFSR are used to create the pseudo-random generator vectors. A generator vector is a bit array of size W which has ΔW ones, and $(1 - \Delta)W$ zeros. By taking $\log_2(W)$ bits in size from the LFSR we have a pseudo-random number with

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3016654, IEEE Transactions on Mobile Computing

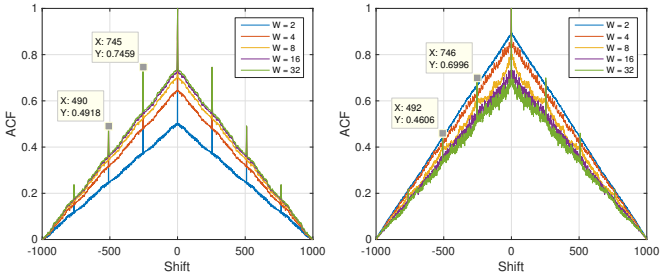


Fig. 10: Normalised autocorrelation functions of output sequences of the pseudo-random functions. (a) Output of the implemented linear feedback shift register. The period of 255 is found in the distance between two peaks in the graph. (b) Output of the implemented pseudo-random generator vector function, with the period of 254.

a maximum value of $W-1$. This can be used as the index of a one in the generator vector. By taking $W\Delta$ distinct numbers from the PRNG a pseudo-random generator vector of size W and degree Δ is generated. To get distinct numbers from the LFSR, for every generation a seed is calculated from the frame sequence number, parity check index and the previous generated pseudo-random number. This does limit the possible values for W to values of the form 2^x.

The autocorrelation of an output sequence of the function generating pseudo-random generator vectors is given in Figure 10(b). To implement the different seeds, the initialisation state of the LFSR is set using a seed as well. Since an initialisation state of 0 does not work, this gives a period of 254, as seen in the figure. The graph lines seen in the autocorrelation function used for the vector generation are coarser, compared to the lines seen in the autocorrelation of the LFSR output. This indicates some form of periodicity and, therefore, non-true randomness in the sequence. However, since the generator vectors will be deployed in a sliding manner, this should not influence the results significantly.

8.2 Decoding Algorithm

We base our decoding strategy from the traditional parity check decoding methods. The decoder works in an iterative way as is explained in the flowchart in Figure 11. When a LoRaWAN frame is not received, the decoder will only know after successfully receiving the next frame that it has lost a frame. If, at the reception of a frame, indeed frames appear to be missing, the parity check(s) from that frame will be used to try to recover the data from lost frames. Each time a frame is received, the data unit in this frame is stored and the relation matrix $\mathbb{H}$ between the received parity checks and the data units is constructed. Known data units are removed from the matrix and Gaussian elimination is performed to remove linear dependence from the matrix. If this reduced matrix has invertible columns, data units can be recovered from the matrix. The relation matrix is again reduced until no more columns can be inverted. If there is a leftover matrix, it is stored for the next iteration. The decoding algorithm has been implemented for code rates of

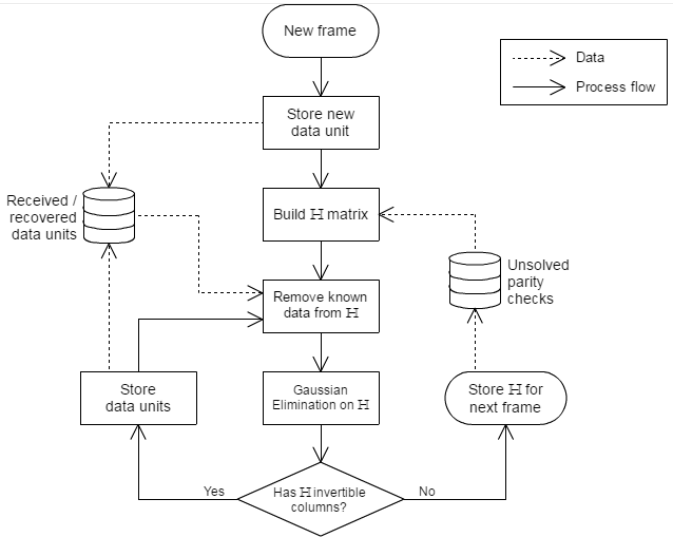


Fig. 11: The flowchart of the decoding algorithm. Data recovery takes place by iteratively solving received parity checks when a frame is received.

R	W	$c[t]$
1 Byte		(k/R) data units

Fig. 12: LoRaWAN payload format for DaRe.

the form $R = 1/x$. The worst case complexity is $O(N^3)$ as Gaussian elimination is the most time consuming operation in decoding.

8.3 LoRaWAN Payload Format for DaRe

The coding parameters R and W used by the encoder can be configured statically in the device. Then the application server, which is the decoder, can have the coding parameters set statically as well, and the coding could work. However, if the device sends the used coding parameters as a header with every frame, it would be possible to change the parameters at runtime.

The used coding parameters will be sent in the payload of every frame along with the coded data, as shown in Figure 12. We implement lookup tables for all possible values of R and W. Therefore, the device only needs to send the indices of the values used in the lookup tables for R and W. Since the number of entries in the tables are limited, we require only four bits each for sending the indices of W and R. The overheads for a frame with k data units would thus be $1 + k(1/R–1)$.

8.4 Choosing and Adapting the Parameters

As explained in Section 6.1, the two main parameters of DaRe, R and W, should be chosen, and Δ is pre-computed with the chosen value for W. R and W are limited to the constraints set by the payload size and end-device memory respectively. By using the results in Figure **??**, the parameter settings providing a desired DRR for the expected frame loss probability and burstiness can be determined. In order

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3016654, IEEE Transactions on Mobile Computing

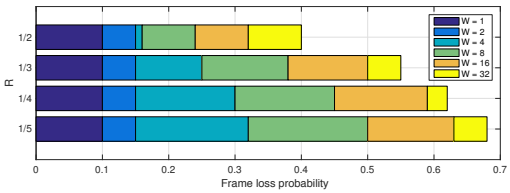


Fig. 13: For a given code rate R the window size needed to recover 99% of the data for a certain frame loss probability.

to make it easy for interpretation, we present the results in Figure 13. Using the expected frame loss, Figure 13 can be used to pick the values for R and W that recovers the expected frame loss up to 99%.

A static choice of parameters may offer varied data recovery performance depending on the long term changes in the frame losses of the channel. One way to adapt these parameters is to use a downlink LoRaWAN frame to notify the end-device about the average channel condition in the recent past. Typical LoRaWAN deployments also know the device locations. The downlink frame may contain the observed average frame losses from the device and nearby devices in the recent past allowing the end-device to dynamically adapt the parameters in run-time to provide optimal performance for the new circumstances.

8.5 System Integration

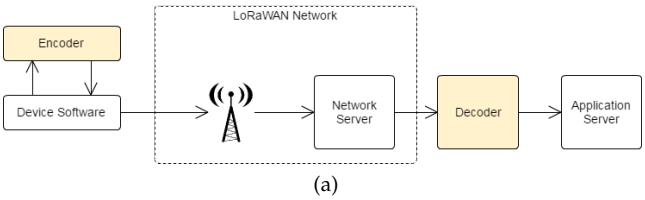
Now all practical challenges have been solved, the encoder and decoder can be implemented in respectively a LoRaWAN end-device and a LoRaWAN application server. In Figure 14(a) a schematic overview of the software architecture for the system integration is given.

The **encoder** is implemented as a C++ library that is included in the device code. The inputs for the encoder function are the coding parameters R and W and the sequence of data units to transmit. The library returns the payload to send in a LoRaWAN frame according to the payload format as shown in Figure 12. To test the library it has been implemented on two devices: the same device used for data collection (Figure 3) and another device with same specifications, which is shown in Figure 14(b).

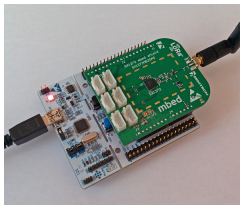
The **decoder** is implemented as a parser between the network server and the application server. The decoder takes frames with DaRe payload and converts them into data units. For emulation the decoder has been implemented as C++ code, but for the final deliverable the decoder has also been implemented in a web server. The web-based decoder is designed to work for multiple devices. The network server only needs to be configured to forward all the frames to the HTTPS endpoint of the decoder, and it will decode for all devices. A screenshot of the interface of the web-based decoder is given in Figure 14(c).

9 Practical Evaluation

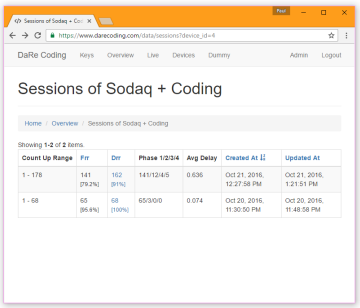
We evaluate our implementation of DaRe on an IID channel that was emulated between the end-device and the application server. We also present results of DaRe using our measurements.



(a)



(b)



(c)

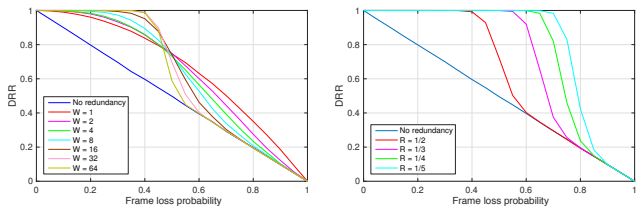
Fig. 14: Implementation. (a) Software architecture. The encoder is implemented as a library that can be included in the device software. The decoder is a module between the Network Server and the Application Server that receives frames as shown in Figure 12 and outputs data units. (b) A second device on which the encoder was tested. (c) A screenshot of the web-based decoder. In the screenshot two sessions of a device using DaRe are shown. The DRR is higher than the FRR as a result of the data recovery by DaRe.

9.1 Evaluation Results

9.1.1 Data Recovery Ratio

Figure 15(a) shows the DRR for various window sizes for code rate R=1/2. It can be seen that the DRR for W=64 is always lower than that for W=32. Thus W=64 does not offer any advantage over W=32. Therefore W=32 is set to be the maximal window size. 99% of the data units can be recovered for channels with frame loss of up to 40%.

For higher frame loss probabilities the value for the code rate should be smaller to recover more data. The DRR for different values of code rate R is shown in Figure 15(b). It can be seen that the DRR increases for lower values of R, as expected. By using a higher code rate, 1/5, we can recover 99% of data with up to 70% frame loss.



(a) (b)

Fig. 15: Data recovery ratio (DRR) in emulation results over an IID channel. (a) DRR for emulation of the decoding algorithm for different window size W and R=1/2. (b) DRR for emulation of the decoding algorithm for different code rates and W=32.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3016654, IEEE Transactions on Mobile Computing

For larger values of W the parity checks contain a larger absolute number of data units, due to the optimal degree we determined before. With a larger number of data units in a parity check the chance of having reducible parity checks is higher for increasing frame loss probability, resulting in lower data recovery. This explains the curves crossing over each other in Figure 15(a).

To reduce the chance for non-reducible parity checks, the degree Δ should decrease. As can be seen in Figure 16, better DRR is reached for higher frame loss probabilities when using a lower degree. However, the results for lower frame loss probabilities will decrease. Since it is more important for the DRR for lower frame loss probabilities to be close to 100%, lower results for higher frame loss probabilities is not relevant. The optimal degree can still be used.

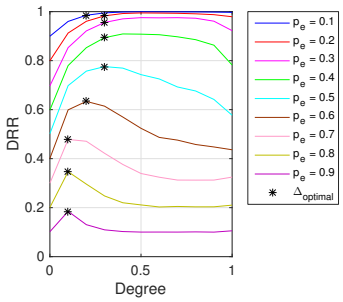
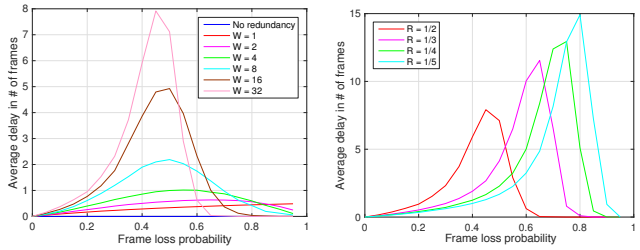


Fig. 16: Data recovery ratio (DRR) for R=1/2 and W=10 for different frame loss probabilities when varying Δ. The black crosses mark the values for the optimal degree $\Delta_{\text{optimal}}(p_e)$ that gives maximal recovery rate. Δ_{optimal} for lower frame loss probabilities is equal to the optimal degree found in Section 6.1.

9.1.2 Data Recovery Delay

Additional to the DRR, we can also determine the average time it takes to decode data now we have a decoding algorithm. The data recovery delay is defined as the number of additional frames needed to be received before a data unit is recovered. Delay could be a factor for a LoRaWAN application, requiring the coding parameters to be selected to minimise delay.



(a) (b)

Fig. 17: Average recovery delay in emulation results of an IID channel. (a) Recovery delay for different window size W and R=1/2. The maximum average delay is 7.9. (b) Recovery delay for different code rates R and W=32.

While larger window sizes result in higher DRR as shown in Figure 15(a), the average delay increases as well, as can be seen in Figure 17(a). A larger frame loss probability introduces a longer delay. At some point, at the point the DRR starts to decrease rapidly, the delay decreases as well. This is due to the fact that some parity checks are left unsolved, reducing the DRR but also the delay. The maximum average delay difference between W=16 and 32 is 3.1 frames. The variance of the delay increases and decreases with the average.

For smaller code rates the top of the average delay graph shifts with the knee point in the DRR graph (Figure 15(b)), as can be seen in Figure 17(b). The maximum average delay increases as well for smaller code rates.

9.2 Measurement Results

To evaluate DaRe for real-life datasets, we have (a) applied DaRe to the previously collected data, and (b) performed some measurements with the end-device running DaRe.

9.2.1 Data Recovery in Collected Data Sets

We have applied DaRe to the datasets collected for the frame loss characterisation. The exact frame losses in these datasets were emulated in emulation setup. This would show the performance of DaRe in real-life datasets without having to redo all data collection. In Figure 18 the DRR is given for five of these dataset emulations. Also the results of one new dataset, collected while using DaRe at runtime, is shown in this figure.

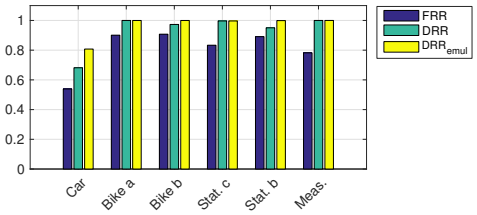


Fig. 18: Data recovery ratio (DRR) in emulation results for five datasets, using parameters R=1/2, W=8. 'Stat.' means stationary data. 'Bike b' and 'Stat. b' are bursty datasets. 'Meas.' is a new dataset acquired while using DaRe at runtime.

We conclude that DaRe performs as expected, where non-bursty datasets have 100% data recovery, and bursty datasets show lower recovery than over an IID channel.

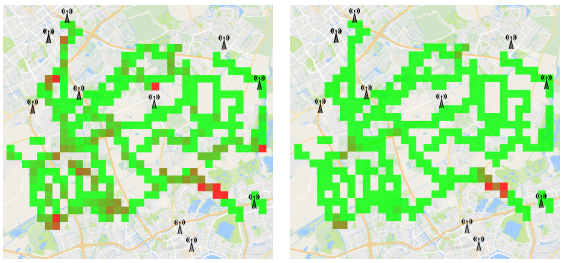
9.2.2 Coverage Improvement

Figure 19(b) shows a map of the coverage of the bike data after applying DaRe with coding parameters R=1/2 and W=8. While there was 14% data loss without DaRe, shown in Figure 19(a), we see that this number reduces to 3% when using DaRe. If a larger window size was used, even more data could be recovered.

9.2.3 Energy

Adding redundancy in communication, like with DaRe, requires transmitting more bytes. The largest contributor to energy consumption on an end-device is the frame transmission. Sending more bytes leads to significantly more energy consumption than additional computations. The impact of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3016654, IEEE Transactions on Mobile Computing



(a) (b)

Fig. 19: Coverage without (a) and with (b) DaRe with R=1/2, W=8. With coding the frame loss is reduced from 14% to 3%. On a colour scale from green to red the frame loss is expressed. Green blocks indicate no loss and red blocks indicate loss of all frames.

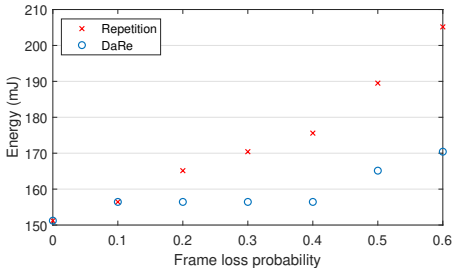


Fig. 20: Comparison of energy consumption of repetition coding and DaRe for achieving 99% DRR for SF7 on an SX1276 radio.

DaRe on the energy consumption of the end-device can be determined by calculating the additional transmission time needed. The airtime of a LoRaWAN frame can be calculated using the formulae given in [51], and the power consumption of SX1276 can be obtained from [41]. Multiplying the power consumption and the airtime yields the energy consumed for the transmission. A detailed energy consumption calculation method is outlined in [52].

Figure 20 shows the energy consumption for both repetition coding and DaRe for SF7 on a SX1276 radio with standard parameters (13 byte header and coding rate of 4/5) for recovering 99% of the data. The energy consumption calculation includes energy consumed for all the phases from the radio waking up to the radio being turned off and with one receive window slot. For a frame loss probability of 0.6, DaRe consumes only 83% of the energy of repetition coding to achieve the same results. While this figure shows energy consumption specifically for SF7, we provide a SF agnostic comparison in Figure 21.

Figure 21 shows the ratio of transmission time for DaRe and repetition coding when 99% data recovery is desired, compared to the transmission time when sending data without coding. DaRe reduces the additional transmission time compared to repetition coding up to 42% for a data unit size of 10 bytes. When no coding is used for frame loss probability of 0, both DaRe and repetition coding takes the same amount of airtime. However, when the frame loss probability increases, redundancy must be added. When redundancy is added, either by repetition or DaRe coding, data can be recovered for a range of frame loss probabilities.

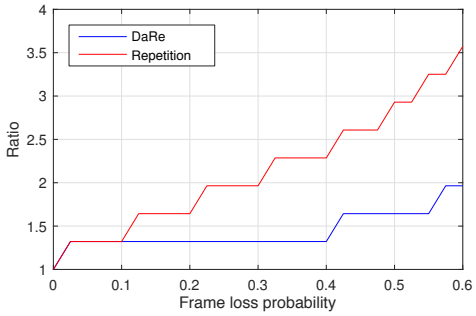


Fig. 21: Transmission time increase for a 10 byte data unit for a desired DRR of 0.99, relative to the transmission time for uncoded data transmission. DaRe provides significant transmission time reduction.

For instance, with R=1/2, and W = 32, DaRe can recover data up to 40% losses as shown in Figure 13. Note that these parameters have been rightly chosen (R=1/2, W=32, and Δ=32). This results in the constant transmission time up to 0.4 in Figure 21. Similarly for repetition coding, the redundancy introduced will be able to recover data for a range of erasure probabilities, which turns out to have smaller flat lines as compared to DaRe. . The parameters were adapted (R=1/3) when the frame loss probability is more than 0.4 to maintain the data recovery ratio at 99%. This increases the overhead and thus, the corresponding increase in the ratio for DaRe. Larger data unit sizes will give even more transmission time reduction.

10 Conclusion

With many IoT applications on the rise, the number of IoT devices is proliferating. To cater to the communication needs of this large number of IoT applications, new architectures and protocols have been proposed recently. One such protocol is LoRaWAN, a Low Power Wide Area Network (LPWAN) technology. While LoRaWAN can provide a large coverage, its basic operating mode is limited by the frame losses due to the unreliable wireless channel and absence of retransmission schemes. In this paper, we characterised frame loss in LoRaWAN. We performed large scale measurements in an almost collision free network deployment for different scenarios. With the datasets, we characterised frame loss in the network in terms of spatial and temporal properties. We found that the frame loss is significant. Furthermore, we demonstrated the frame loss can be bursty in nature even for stationary end-devices.

Conventional wireless techniques, such as using ACK for every transmitted frame, are withheld in LoRaWAN to provide scalability: and to save transmission time on the gateways and also energy on the end-devices. Thus, it is up to the application layer to guarantee and increase the data reception. To this end, we propose a novel erasure coding method, DaRe, that reduces data loss in LoRaWAN significantly. DaRe is based on applying fountain codes on a sliding window. To simulate DaRe we described an algebraic framework. We evaluated DaRe both with emulations and by implementing it on an end-device. We achieved significant recovery of 99% with a code rate R=1/2

when channel erasure probability was 0.4. Comparing with a naive repetition coding method, DaRe reduces energy requirement up to a factor of 0.42. Further, we showed that DaRe can lower an average data loss of 14% in 300 km of biking to 3% data loss with R=1/2 and W=8.

Since the network is newly deployed, collisions are negligible. When more devices start connecting to the network collisions will start contributing significantly to the overall frame loss in the network. There will be an increased need for data recovery methods that do not add much load to the network. Therefore, we believe that DaRe provides a sustainable solution to improve the data throughput.

The two most important open problems are to analytically derive optimal degree and the protocol that integrates with ADR. Furthermore, the impact of DaRe with more number of devices need to be studied in realistic scenarios, which are part of our future work.

ACKNOWLEDGMENT

This research was partially carried out within the SCOTT project (http://www.scott-project.eu) funded from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737422. This joint undertaking receives support from the European Union's Horizon 2020 research and innovation program and Austria, Spain, Finland, Ireland, Sweden, Germany, Poland, Portugal, Netherlands, Belgium, Norway.

REFERENCES

[1] P. Marcelis, V. Rao, and R. V. Prasad, "DaRe: Data Recovery through Application Layer Coding for LoRaWAN," in Proceedings of the Second International Conference on Internet-of-Things Design and Implementation. ACM, 2017, pp. 97–108.

[2] J. Bartje, "The top 10 IoT application areas – based on real IoT projects," https://iot-analytics.com/top-10-iot-project-application-areas-q3-2016/, accessed: 2016-01-13.

[3] M. Bor, J. Vidler, and U. Roedig, "LoRa for the Internet of Things," in Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks. Junction Publishing, 2016, pp. 361–366.

[4] 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.

[5] N. Sornin, M. Luis, T. Eirich, T. Kramp, and O. Hersent, "LoRaWAN™ Specification," https://www.lora-alliance.org/portals/0/specs/LoRaWAN%20Specification%201R0.pdf, Jan 2015.

[6] Semtech, "LoRa™ Modulation Basics," https://semtech.my.salesforce.com/sfc/p/E0000000JelG/a/2R0000001OJk/yDEcfAkD9qEz6oG3PJryoHKas3UMsMDa3TFqz1UQOkM, May 2015.

[7] R. Sanchez-Iborra, J. Sanchez-Gomez, J. Ballesta-Viñas, M.-D. Cano, and A. F. Skarmeta, "Performance Evaluation of LoRa Considering Scenario Conditions," Sensors, vol. 18, no. 3, p. 772, 2018.

[8] D. Zorbas, K. Abdelfadeel, P. Kotzanikolaou, and D. Pesch, "TS-LoRa: Time-slotted LoRaWAN for the Industrial Internet of Things," Computer Communications, vol. 153, pp. 1–10, 2020.

[9] T. Polonelli, D. Brunelli, A. Marzocchi, and L. Benini, "Slotted aloha on LoRaWan-design, analysis, and deployment," Sensors, vol. 19, no. 4, p. 838, 2019.

[10] 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," International Journal of Distributed Sensor Networks, vol. 13, no. 3, p. 1550147717699412, 2017.

[11] M. Aref and A. Sikora, "Free space range measurements with Semtech LoRa technology," in 2014 (IDAACS-SWS), Sept 2014, pp. 19–23.

[12] J. Petajajarvi, K. Mikhaylov, M. Hamalainen, and J. Iinatti, "Evaluation of LoRa LPWAN technology for remote health and wellbeing monitoring," in 2016 ISMICT, March 2016, pp. 1–5.

[13] J. Petajajarvi, K. Mikhaylov, A. Roivainen, T. Hanninen, and M. Pettissalo, "On the coverage of LPWANs: range evaluation and channel attenuation model for LoRa technology," in ITS Telecommunications (ITST), 2015 14th International Conference on, Dec 2015, pp. 55–59.

[14] K. Mikhaylov, J. Petajajarvi, and T. Hanninen, "Analysis of Capacity and Scalability of the LoRa Low Power Wide Area Network Technology," in European Wireless 2016; 22th European Wireless Conference, May 2016, pp. 1–6.

[15] A. Augustin, J. Yi, T. Clausen, and W. Townsley, "A Study of LoRa: Long Range & Low Power Networks for the Internet of Things," Sensors, vol. 16, no. 9, p. 1466, Sep 2016.

[16] M. Bor, U. Roedig, T. Voigt, and J. Alonso, "Do LoRa low-power wide-area networks scale?" in The 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, 2016.

[17] N. Rathod, P. Jain, R. Subramanian, S. Yawalkar, M. Sunkenapally, B. Amrutur, and R. Sundaresan, "Performance analysis of wireless devices for a campus-wide iot network," in 2015 13th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt). IEEE, 2015, pp. 84–89.

[18] J. Ababneh and O. Almomani, "Survey of Error Correction Mechanisms for Video Streaming over the Internet," (IJACSA) International Journal of Advanced Computer Science and Applications, vol. 5, no. 3, 2014.

[19] X.-J. Zhang and X.-H. Peng, "A testbed of erasure coding on video streaming system over lossy networks," in Communications and Information Technologies, 2007. ISCIT '07. International Symposium on, Oct 2007, pp. 535–540.

[20] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," Journal of the Society for Industrial and Applied Mathematics, vol. 8, no. 2, pp. 300–304, 1960. [Online]. Available: http://www.jstor.org/stable/2098968

[21] M. Mitzenmacher, "Digital fountains: a survey and look forward," in Information Theory Workshop, 2004. IEEE, Oct 2004, pp. 271–276.

[22] R. Gallager, "Low-density parity-check codes," IRE Transactions on information theory, vol. 8, no. 1, pp. 21–28, 1962.

[23] A. Morello and V. Mignone, "DVB-S2: The Second Generation Standard for Satellite Broad-Band Services," Proceedings of the IEEE, vol. 94, no. 1, pp. 210–227, Jan 2006.

[24] J. Qureshi, C. Heng Foh, and J. Cai, "Primer and recent developments on fountain codes," Recent Advances in Communications and Networking Technology (Formerly Recent Patents on Telecommunication), vol. 2, no. 1, pp. 2–11, 2013.

[25] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," IEEE Transactions on Information Theory, vol. 47, no. 2, pp. 569–584, Feb 2001.

[26] M. Luby, "LT codes," in Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on, 2002, pp. 271–280.

[27] A. Shokrollahi, "Raptor codes," IEEE Transactions on Information Theory, vol. 52, no. 6, pp. 2551–2567, June 2006.

[28] P. Maymounkov, "Online codes," Technical report, New York University, November 2002. [Online]. Available: https://pdos.csail.mit.edu/~petar/papers/maymounkov-online.pdf

[29] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1," in Communications, 1993. ICC, IEEE International Conference on, vol. 2, May 1993, pp. 1064–1070 vol.2.

[30] M. Usman, "Convolutional fountain distribution over fading wireless channels," International Journal of Electronics, vol. 99, no. 8, pp. 1037–1050, 2012.

[31] H. Jenkac, J. Hagenauer, and T. Mayer, "The Turbo-Fountain and its Application to Reliable Wireless Broadcast," in Wireless Conference 2005 - Next Generation Wireless and Mobile Communications and Services (European Wireless), 11th European, April 2005, pp. 1–7.

[32] C. Studholme and I. F. Blake, "Random Matrices and Codes for the Erasure Channel," Algorithmica, vol. 56, no. 4, pp. 605–620, 2010. [Online]. Available: http://dx.doi.org/10.1007/s00453-008-9192-0

[33] M. Sandell and U. Raza, "Application layer coding for iot: benefits, limitations, and implementation aspects," IEEE Systems Journal, vol. 13, no. 1, pp. 554–561, 2018.
[34] Z. Zhao, W. Dong, G. Chen, G. Min, T. Gu, and J. Bu, "Embracing corruption burstiness: Fast error recovery for zigbee under wi-fi interference," IEEE Transactions on Mobile Computing, vol. 16, no. 9, pp. 2518–2530, 2016.
[35] C. Noda, S. Prabh, M. Alves, and T. Voigt, "On packet size and error correction optimisations in low-power wireless networks," in 2013 IEEE International Conference on Sensing, Communications and Networking (SECON). IEEE, 2013, pp. 212–220.
[36] C.-J. M. Liang, N. B. Priyantha, J. Liu, and A. Terzis, "Surviving wi-fi interference in low power zigbee networks," in Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems. ACM, 2010, pp. 309–322.
[37] K. Ishibashi, H. Ochiai, and R. Kohno, "Embedded forward error control technique (efect) for low-rate real-time communications," in IEEE GLOBECOM 2007-IEEE Global Telecommunications Conference. IEEE, 2007, pp. 4565–4569.
[38] A. Das, A. Ghose, A. Razdan, H. Saran, and R. Shorey, "Enhancing performance of asynchronous data traffic over the bluetooth wireless ad-hoc network," in Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213), vol. 1. IEEE, 2001, pp. 591–600.
[39] Texas Instruments, "ISM-Band and Short Range Device Regulatory Compliance Overview," http://www.ti.com/lit/an/swra048/swra048.pdf, May 2005.
[40] X. Xia, Y. Zheng, and T. Gu, "Ftrack: parallel decoding for lora transmissions," in Proceedings of the 17th Conference on Embedded Networked Sensor Systems. ACM, 2019, pp. 192–204.
[41] Sodaq, "Sodaq Mbili," http://support.sodaq.com/sodaq-one/mbili/schema-rev-4/, accessed: 2016-10-09.
[42] Semtech, "ETSI Compliance of the SX1272/3 LoRa Modem," http://www.semtech.com/images/datasheet/etsi-compliance-sx1272-lora-modem.pdf, July 2013.
[43] Y. Huang and K. Boyle, Antennas: from theory to practice. John Wiley & Sons, 2008.
[44] A. Goldsmith, Wireless communications. Cambridge university press, 2005.
[45] S. Demetri, M. Zúñiga, G. P. Picco, F. Kuipers, L. Bruzzone, and T. Telkamp, "Automated estimation of link quality for lora: a remote sensing approach," in 2019 18th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN). IEEE, 2019, pp. 145–156.
[46] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," IEEE/ACM Transactions on Networking, vol. 2, no. 1, pp. 1–15, Feb 1994.
[47] E. N. Gilbert, "Capacity of a burst-noise channel," The Bell System Technical Journal, vol. 39, no. 5, pp. 1253–1265, Sept 1960.
[48] H. Bai and M. Atiquzzaman, "Error modeling schemes for fading channels in wireless communications: A survey," IEEE Communications Surveys & Tutorials, vol. 5, no. 2, 2003.
[49] M. Zorzi and R. R. Rao, "Lateness probability of a retransmission scheme for error control on a two-state markov channel," IEEE Transactions on Communications, vol. 47, no. 10, pp. 1537–1548, 1999.
[50] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, "Numerical recipes in c++," The art of scientific computing, vol. 2, p. 1002, 1992.
[51] Semtech, "SX1272/73 - 860 MHz to 1020 MHz Low Power Long Range Transceiver," March 2015.
[52] T. Bouguera, J.-F. Diouris, J.-J. Chaillout, R. Jaouadi, and G. Andrieux, "Energy consumption model for sensor nodes based on lora and lorawan," Sensors, vol. 18, no. 7, p. 2104, 2018.