Fragmentation and Forward Error Correction for LoRaWAN small MTU networks

Ulysse Coutaud, Martin Heusse, Bernard Tourancheau

Fragmentation and Forward Error Correction for LoRaWAN small MTU networks

Ulysse COUTAUD
Semtech Corporation and
LIG CNRS,
Grenoble Alps University
first.last@univ-grenoble-alpes.fr

Martin HEUSSE
LIG CNRS, Grenoble INP
first.last@imag.fr

Bernard TOURANCHEAU
LIG CNRS,
Grenoble Alps University
first.last@imag.fr

Abstract

We present the LoRaFFEC protocol, which proposes fragmentation and Forward Error Correction to the LoRaWAN™ standard. The idea of this new mechanisms is to make the applications worldwide compatible with all LoRaWAN™ deployments, with a similar sensitivity to packet losses. With LoRaFFEC, any payload length can benefit from better Data Delivery Rate (DDR) for a wide range of channel-level packet losses.

I. Introduction

LoRaWAN™ operates in the license free Industrial Scientific and Medical (ISM) bands which are subject to legal restrictions: in particular, the maximum transmission time duty cycle is often very limited. The regulation varies between frequency bands and countries. Moreover, LoRaWAN™ dynamically adjusts the Data Rate (DR) to adapt to the channel conditions, resulting in different channel occupation. As a consequence the available application payloads varies, from 11 bytes to a maximum of 250 bytes. Fragmentation is thus needed to ensure application compatibility with all legislations and any link quality.

As any wireless technology, LoRaWAN™ is subject to packet loss. Its Packet Error Rate (PER) is variable due to the electromagnetic environment, limited emission power, network load and channel gain variability . Some intra-packet Forward Error Correction (FEC) is present as part of the LoRaWAN™ physical layer. However, a better application Data Delivery Rate (DDR) can be obtained by adding cross-packet level FEC.

In this paper we introduce a mechanism to solve both challenges of fragmentation and application level FEC, resulting in better DDR without payload length limitation issues in LoRaWAN™ IoT applications.

This article is organized as follow: Section II presents the main characteristics of LoRaWAN™ technologies. Section III describes previous works on LoRa™ channel characterization and field measurements as well as on FEC in wireless networks. Section IV presents LoRaFFEC, the protocol designed to solve the fragmentation and FEC challenges. Section V gives simulated performances with a sensitivity analysis and fragmentation impact.

II. LoRaWAN™

LoRaWAN™ defines a communication protocol and system architecture based on the LoRa™ physical layer which enables long-range and low power communication. LoRaWAN™ is a cellular network in that always-on gateways relay the cyphered messages between the numerous battery-powered end-devices to the network server which then routes the messages to the application servers. End-devices are not associated with a specific gateway but rather, macro diversity is key to obtain good data delivery: a frame emitted by an end-device is typically received by multiple gateways which forward it to the network server where deduplication takes place. Most of the operations are pushed to the network server, which

TABLE I: Application payload (without any optional fields in the header) for the minimum and maximum DR in various bands.

Band	Maximum payload size (bytes)	
	Min DR	Max DR
EU863-870	51	242
US902-928	11	242
CN779-787	51	242
EU433	51	242
AU915-928	11	242
CN470-510	51	242
AS923	19	250
KR920-923	51	242
IN865-867	51	242
RU864-870	51	242

manages the network, filters duplicates, performs security checks, schedules acknowledgments through the appropriate gateway, and determines and issues LoRaWAN™'s Adaptive Data Rate (ADR) protocol commands to the nodes. If an end-device is mobile or moving, there is no handover needed from gateway to gateway[1]. LoRaWAN™ MAC channel access is a simple Aloha [1] scheme. The frame structure is composed of the synchronization preamble, header, payload and CRC. Apart from operator bands, in Europe, LoRaWAN™ operates mostly in the 863-870 MHz and 433-434.8 MHz ISM bands, representing 70 125 kHz bandwidth channels. In most cases, ISM bands regulation imposes a maximum 1% transmission duty cycle to both end-devices and gateways when using an Aloha access method. Notice that this puts a significant limitation to the downlink capacity of the gateways as they need to serve all the surrounding end-devices with relatively scarce transmission opportunities [2].

The LoRa™ physical layer is a proprietary Chirp Spread Spectrum modulation which trades data rate (DR) for sensitivity within a fixed channel bandwidth. The transmissions at different LoRa™ DRs are quasi-orthogonal. Each DR correspond to a specific spreading factor (SF) numbered from 6 to 12. The intra-packet FEC uses coding rates (CR) 4/5 to 4/8. LoRaWAN™ uses LoRa™ SF7 to SF12 and a fixed intra-packet CR 4/5 FEC. As transmissions on different SFs do not interfere with each other, the available DRs create a number of sub-channels, which increases the capacity of the network [16]. LoRaWAN™ parameters include frame emission repetition, from 1 to 15 times, and transmission power adjustment (P_{Tx}) from 0dBm to 14dBm.

The maximum payload size changes depending on the DR and local regulation. In table I we list the minimal and maximal payload allowed by regulation [10]. Moreover, a part of the payload available for the application data will sometimes be used to transport MAC commands. They are often piggybacked to the LoRaWAN™ payload to perform network management, which in turn modifies and reduces the maximum payload size.

As a consequence, between SF selection, number of repetitions and transmission power (P_{Tx}) adjustments, a LoRaWAN™ system compromises between DR, frame Time On Air, communication range, power consumption, interference level, and network capacity performances, within a constant bandwidth [5].

III. Related works and background

The LoRa™ communication channel is now well characterized. Line-of-Sight communication is possible at a distance of up to 15 km on land and 30 km on sea with P_{Tx} 14 dBm and SF12, although communication at this range has a high PER: respectively 74%

[1]This is a critical feature to enable asset tracking applications, a major target application for IoT [9]

and 38% [13]. In another study [12], researchers from the same research group measure the Received Signal Strength Indicator (RSSI) and PER, in an indoor environment for a range from 50 m to 400 m. In this case, they observe a PER around 3.3% for a stationary device and 5% for a moving one, showing that a residual PER is to be expected for indoor communications. Other experimental measures of LoRa™ RSSI and PER in various radio environments: open area, straight road, moderately wooded area, highly wooded area, at distances between 50 to 1000 m show low PER: less than 0.8% at SF7 and P_{Tx} =5 dBm in all environments except for the highly wooded area where the PER is 40% at 250 m [15]. But other tests and measurements of LoRa™ Signal to Noise Ratio quality (SNR) for both the 433MHz and 868MHz bands, in both indoor and outdoor situations, show that the link is highly variable: a rain fall, the presence of people into the building impact the received signal strength and stability [3]. Radio link quality is only one aspect of the performance in a LoRaWAN™ cell: Augustin et al. estimated collision rate as well as the attainable throughput, the experimental coverage and quality of the LoRa™ link in an urban scenario [5]. Their measures show that the PER smoothly increases from 0 to 100% with the distance. Marcellis et al. [11] also performed experiment over stationary and mobile LoRa™ devices. They observed that the PER over the LoRa™ channel can be uniform or bursty, even for stationary devices. They propose to model the erasures over the channel either with a Bernoulli or a Gilbert Elliot channel model. In fact, another study [4] concludes that the initial reception condition strongly affects each frame reception, reducing the payload size influence on the frame reception. They also characterize the channel as a Rayleigh fading channel. These studies show that even if long range transmission can be achieved, LoRaWAN™ deployment is subject to significant PER with a residual erasure rate even in favorable conditions.

FEC consists in encoding an M symbols long dataword into a codeword using an Error Correction Code (ECC) which introduces N symbols of redundancy. The dataword can thus be recovered from the codeword as long as the fraction of erased or corrupted symbols is less than respectively e and c, which are usual characteristics of a given ECC. FEC is broadly used in satellite communications and telecommunications to improve communication reliability and throughput. In order to match the various requirements of the applications (error/erasure correction capacity, computation/memory constraints) or to match the nature of the channel model (Gaussian noise, burst error, frame loss, byte error,...) various ECC have been developed and deployed, such as Turbo code [6] in UMTS 3G, onvolutional code [17] in GSM 2G or Low-Density Parity-Check (LDPC) [8] in DVB-S2.

LoRa™ itself already uses FEC at the intra-packet level as described in section II. This improves the demodulation capacity by allowing a gateway to correctly receive a frame even if a part of it is mis-demodulated. The Coding Adaptive Redundancy Rate (CCARR) scheme is an example of inter-packet FEC [7]. CCARR uses Reed-Solomon coding to implement inter-frame FEC at the application layer of LoRaWAN™. CCARR sharply boosts the acceptable PER while reducing the channel load by dynamically tuning the effective coding rate. But CCARR relies on regular downlinks which should be kept as rare as possible in the LoRa™ context, as discussed in several works [14], [2]. Marcellis et al. [11] propose another inter-frame FEC protocol, Data Recovery (DaRe) based on LDPC codes which doesn't require any downlinks. However, their experiments do not corroborate their simulation results which questions one or the other, model or implementation. In their work, the FEC processing is not interleaved with any fragmentation but associated to each fixed size data fragment which limits the granularity and flexibility of the approach. However if the FEC scheme of DaRe [12] is very similar to the one of LoRaFFEC, there are inconsistencies in their evaluation and a lack of implementation details which precludes the research results repetition. So we believe, as described in the following, that our independent implementation and performance gain assessment clarifies the interest of such a solution.

Moreover, we emphasize that it is the combination of FEC and fragmentation which brings added benefits. In particular, it makes our solution pertinent for industrial deployment because - On the one hand, fragmentation requires an error recovery scheme since the loss of one fragment leads to discarding the whole Application Data Unit; - On the other hand, the redundant data overhead inherent to FEC reinforces the need for fragmentation. Thus, overall, fragmentation and FEC have a cumulative effect on our metric of interest, the Data Delivery Rate.

Similarly to them, our solution LoRa Fragmentation and FEC (LoRaFFEC) uses LDPC encoding, without downlink communication. It combines error correcting with a fragmentation mechanism, thus addressing the varying payload length issues of LoRaWAN™. This allows to solve two issues while offering a solution which is appropriate given the diversity of transmission parameters of LoRaWAN™.

IV. LoRaFFEC protocol and its parameters

LoRaFFEC considers uplink communications between a device and the server as a stream of packets which need to abide by the regulation of the considered frequency band. The stream implements LDPC-like FEC by computing redundancy packets on the fly using pseudo-random linear combinations of already sent data. On top of that mechanism, application data is fragmented to fulfill the stream packet size.

Thus LoRaFFEC implements the fragmentation of the application payload, called Application Data Unit (ADU) and thanks to the stream and FEC created, it allows the recovery of lost data. On the one hand, fragmentation has a strong negative impact on DDR: every single fragment of an ADU needs to be received for reconstructing the original ADU, thus, DDR = (Fragment_ Reception_Rate)$^{\#\text{fragments}}$. So it is natural to supplement a fragmentation scheme with a mechanism such as inter-frame FEC to regain a satisfactory DDR . On the other hand, FEC implies more payload overhead to carry the additional redundancy control information. This reinforces the need for fragmentation, in order to be able to transmit any payload size, by splitting if necessary the data over multiple packets while optimizing the use of each packet transmission with the aggregation of many fragments when possible. This interdependence affirms the need for a protocol ensuring both fragmentation and Quality of Service in terms of DDR.

LoRaFFEC is realized through three sub-layers performing respectively: integrity check, fragmentation and erasure correction. A LoRaFFEC device first sends data fragments, followed by a number of redundancy fragments as shown in Fig.1. Thus, the device has to store a window of previous fragments. On the server side, when a data fragment is missing, the previously received data and redundancy fragments form a linear system of equations which enables to attempt to rebuild the data. This involves storage and relatively heavy computation. The corresponding device and server LoRaFFEC algorithms are presented in Section IV-E and IV-F with a precise evaluation of the memory and computational complexity costs.

To precisely assess LoRaFFEC, several parameters have to be discussed: the window length (w) on which the FEC is computed, the redundancy density (RD) which is the number of data fragments combined in each redundancy one, and the decoding depth (DD) that the server considers when building the system of equations.

A. Erasure correction sub-layer

The erasure correction sub-layer implements the FEC techniques to recover the data lost during communication. The correction code is a derivative of LDPC [8] and we call it Multiple Parity Check (MPC). MPC coding consists in sending, in addition to the data fragments, pseudo-random linear combinations of the data fragments written in redundancy fragments. Eventually, lost data fragments might be recovered from solving the linear system formed of the successfully received data and redundancy fragments.

B. Window length

The data fragments included in the combinations are pseudo-randomly chosen among a window of the previous data fragments. This window spans between fragment$[i]$ (the current ADU last data fragment) and fragment$[i - w + 1]$, where w is the window length. Redundancy fragments are sent after the data ones (i.e. the data fragments of a single ADU are sent consecutively and cannot be interleaved with redundancy fragments.

In our implementation the fragments are numbered on one Byte. These redundancy fragments are numbered with an offset of 128, relatively to their matching data fragments numbered from 0 to 127. Fig. 1 illustrates the computation windows used for consecutive redundancy fragments. The maximal w is limited by the fragment numbering, here $w \leq 128$. However, device memory requirement grows linearly with w as the device buffer size is $w \times$ fragment_length.

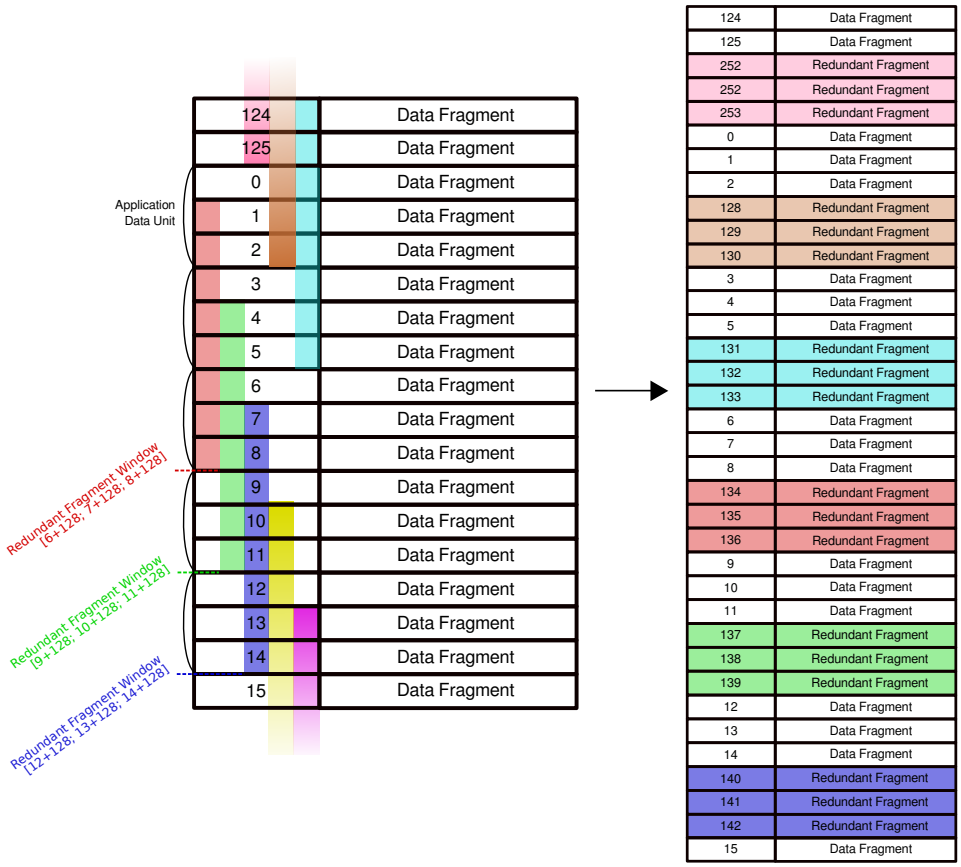

Fig. 1: Fragment computation window and numbering, $w = 8$ and each ADU contains 3 fragments.

Depending on the maximum frame size, each fragment can either be transmitted alone as a LoRaWAN™ payload, or aggregated with other fragments into a multi-fragments LoRaWAN™ payload as illustrated respectively in Fig. 2 and Fig. 3. In the second case, so as to reduce overhead, only one fragment counter is transmitted over the air, the others ones are recomputed at the server side.

Data fragments and redundancy fragments can either be sent separately or together with a LoRaWAN™ packet. In order to allow puncturing of the redundancy and still being able to retrieve correctly each fragment number, data fragments are always sent in front. I.e.

Fig. 2: A LoRaWAN™ payload with a single fragment.

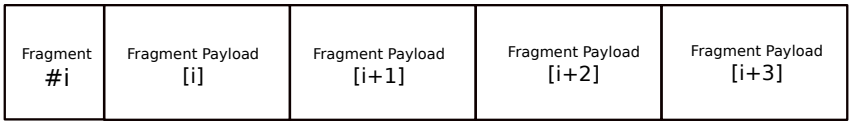

Fig. 3: A LoRaWAN™ payload with multiple fragments.

data fragments of a new ADU cannot be piggybacked after the redundancy fragments of the next ADU. Fig. 4 illustrates the aggregations possibilities.

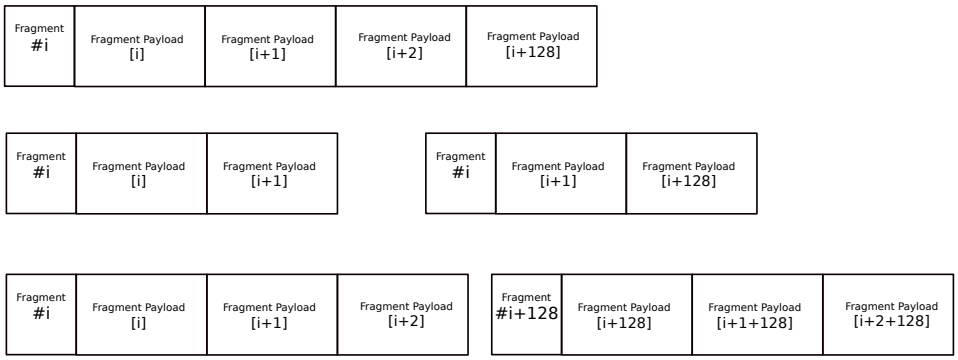

Fig. 4: Aggregation of data and redundancy fragments into LoRaWAN™ frames.

C. Redundancy density

The redundancy density RD represents the number of data fragments combined to constitute a redundancy fragment as a proportion of the w. Thus, with RD $= 0$ no redundancy fragment is computable, whereas with RD $= 1$ all data fragments in the window are always combined in each redundancy. For instance, in LoRaFFEC with $w = 128$ and RD $= 0.5$, 64 pseudo-randomly chosen data fragments are combined into each redundancy fragment. Hence, RD influences the amount of computation on the devices. Notice that the server and the devices share the same pseudo-random generator (with the same seed) for use in the FEC process.

D. Decoding depth

The decoding of LoRaFFEC consists in progressively diagonalizing a binary index matrix in which each fragment corresponds to a line. In this index line, each "1" corresponds to a data fragment combined into the fragment. Thus, a data fragment is a line comprising a single "1", whereas a redundancy fragment is a line with multiple "1"s. Each new line in the index matrix can, by linear combination gear effect, resolve a data fragment from the past. There is no bound to this effect, so a missing data fragment could, in theory, be unlocked infinitely long after its reception as part of a redundancy fragment. However, the practical bound is the storage and the computation available. In order to make the decoding sustainable, we bounded the backtracking DD as a multiple of w.

E. Encoding on the device

On the device, following the algorithm described in Algo. 1, the encoding process computation complexity for each redundancy fragment is $w \times \mathrm{RD}$. The real bottleneck is the memory footprint: the device needs a buffer of $w \times f$ bytes, where f is the fragment length, to store the past ADUs that are used to compute the current redundancy fragments.

```
sequence ← Random(fragment#);
forall  j ∈ sequence do
 |  redundantfragment ← LinearCombination(w,RD,fragment[j]);
end
Transmit(redundantfragment);
```

Algorithm 1: Fragmentation and redundancy fragment encoding algorithm using LDPC.

F. Decoding on the server

The decoding is computed using a best effort linear resolution of the LDPC encoding. The work matrix lines are filled with all the received fragments indexes followed by their data. The linear system solving operations are determined and computed from the indexes structure. All the corresponding (XOR) operations are applied to the whole data part of the fragment in order to reconstruct the lost elements. When a new fragment comes in, diagonalisation of the index matrix is advanced as much as possible using Gaussian elimination on the new fragment and then with the fragment itself and possible new pivot(s). In the end, each updated index line which only contains one "1" has its data part delivered to the de-fragmentation procedure. Otherwise, it stays as is, waiting for the next fragment diagonalisation to be applied for possible complete decoding later on.

```
forall newfragment do
 |  StoreData(newfragment,datamatrix);
 |  InsertInDiagonal(newfragment,indexmatrix);
 |  DiagonalUpdates(indexmatrix,datamatrix,DD);
 |  if LinesWithSingleOne(indexmatrix) then
 |   |  DeliverDataFragments(datamatrix);
 |  end
end
```

Algorithm 2: Decoding and update algorithm of the index and data matrices.

At the server side, from the algorithm presented in Algo. 2, the decoding process computation complexity is $O(w^2 \times \mathrm{DD})$ for each received fragment and it requires a buffer of size $(\mathrm{DD} \times w)^2/8 + (\mathrm{DD} \times w \times f)$ bytes in memory to store the decoding matrices. With the discussed practical numbers, this stays very reasonable for a server.

V. LoRaFFEC simulation performance

We performed extensive simulations to evaluate the performance of LoRaFFEC with parameter variations. The simulation scenario decodes a LoRaFFEC stream with random packet erasure. Erasures are independent and identically distributed.

A. Simulator description

To evaluate the LoRaFFEC protocol and check the proposed FEC, a simulator was designed which executes the actual LoRaFFEC protocol on a simulated communication channel. The simulator encodes the input stream with LoRaFFEC, and pushes it to

the LoRaFFEC decoder engine following the pre-computed channel erasure pattern. The channel erasure pattern simulates the wireless communications using a Raleigh channel model following the i.i.d. assumptions. The channel parameters are set to mimic LoRa™ physical and LoRaWAN™ link layers.

B. Simulation results

LoRaFFEC has various parameters that can be tuned to maximize its data delivery rate and latency: w and RD at the encoding side and DD at the decoding side. We note LoRaFFEC (x, y, z) a LoRaFFEC experiment with $w = x$, RD $= y$ and DD $= z$. In Fig. 5 we compare LoRaFFEC (w,0.6,5) performance for $w \in \{8, 16, 32, 128\}$. Data fragments retransmission (each data fragment is transmitted twice) results are also plotted for the sake of comparison. LoRaFFEC recovers almost every erased data over a channel up to 40% erasure rate. The DDR stays above 98% with channel erasure rate $< 30\%$ for $w = 8$, $< 40\%$ for $w = 16$ and 45% for $w = 32$ and 128. When the channel quality gets worse, data delivery drops in a waterfall manner down to the channel erasure rate, meaning that LoRaFFEC is unable to recover any lost data. LoRaFFEC with large w offers more diversity and thus is less sensitive to channel erasure rate local variations. But the waterfall is more pronounced for longer w, i.e. DDR stays $> 98\%$ further to the right, but then it drops at a steeper angle.

As we target to offer high reliability in practice, LoRaFFEC with $w = 128$ should deliver more than 98% of the data over lossy channels up to PER $= 45\%$. For higher data loss on the channel, others solutions like LoRaWAN™ packets retransmissions or ADR downlink commands for higher transmit power or switching of SF, must be considered maintain proper functioning. Fig. 6 shows LoRaFFEC Data Delivery Rate (DDR) progression for various redundancy density when transmitting on a 40% erasure rate channel. For small w, the RD has some influence, has shown on Fig. 6 (a) but for larger w, the DDR curves stays very close from each other before the waterfall. Hence, in our $w = 128$ target implementation we chose RD ≥ 0.25. Fig. 7 shows LoRaFFEC (128,0.6,DD) recovery performances for DD $\in \{1, 2, 3, 4, 5\} \times w$. The curves presents a good performance flat part followed by a step waterfall-like degradation and a higher DD pushes the waterfall to the right. However, in practice, the only significant variation happens with $DD = 1 \times w$. This corresponds to the situation where the decoding matrix is barely diagonalizable, i.e. some erased data fragments are still present into redundancy fragments but could not be extracted. As $DD > 2 \times w$, the variation is very small; in our implementation we set DD $= 2 \times w$ to keep decoding complexity down.

VI. Conclusion

LoRaFFEC proposes to combine fragmentation and application-level error correction, to reach satisfactory data delivery DDR>98%, for up to 40% PER channels using LDPC encoding. This is demonstrated over simulated channels. However, LoRaFFEC performance drops for larger channel PER. For a channel PER>40%, FEC redundancy cannot be exploited and its overhead is wasted. As a consequence, in order to provide almost perfect DDR, LoRaFFEC very efficient gain need to be integrated within dynamic channel tuning - such as the LoRaWAN™ ADR - to maintain the communication channel PER $< 40\%$. Current LoRaWAN™ networks usually waste channel capacity by using conservative parameters such as high SF and P_{Tx}, as well as systematic LoRa™ retransmissions in order to ensure an acceptable DDR. A smart usage of ADR alongside with LoRaFFEC will allow to relax these parameters and lead to substantial improvement in network capacity while ensuring very high DDR.

Our future work targets a deepest understanding of the LoRa™ channel erasure patterns and the integration of LoRaFFEC into the LoRaWAN™ ADR.

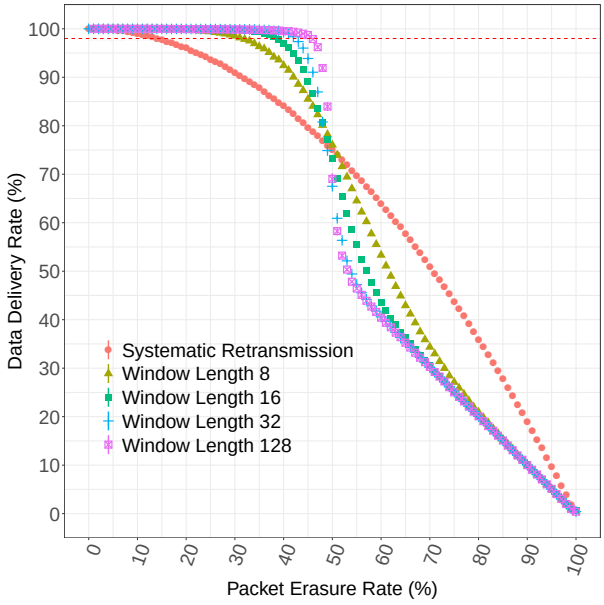

Fig. 5: Simulated DDR as a function of PER with LoRaFFEC $(w; 0.6; 5)$.

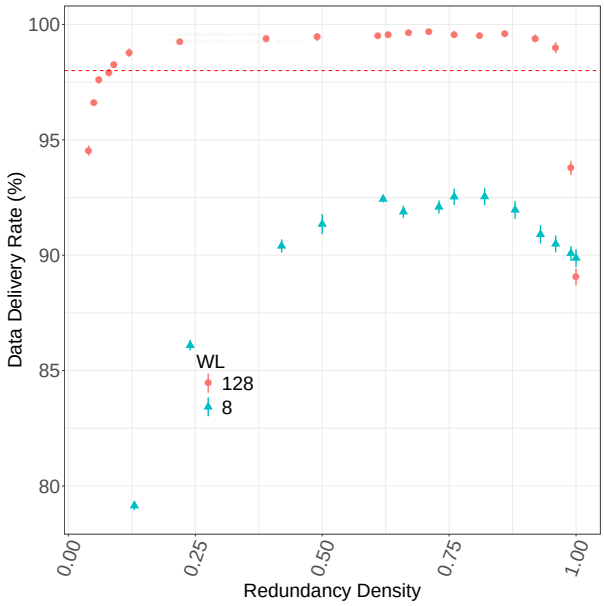

Fig. 6: Simulated DDR as a function of RD with LoRaFFEC $(128; \mathrm{RD}; 2)$ and LoRaFFEC $(8; \mathrm{RD}; 2)$ over a 40% erasure channel.

References

[1] N. Abramson. Development of the alohanet. IEEE Transactions on Information Theory, 31(2):119–123, Sept. 2006.

[2] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia, and T. Watteyne. Understanding the limits of lorawan. IEEE Communications magazine, 55(9):34–40, 2017.

[3] T. Ameloot, P. Van Torre, and H. Rogier. A compact low-power LoRa IoT sensor node with extended dynamic range for channel measurements. MDPI Sensors, 18(7):2137, 2018.

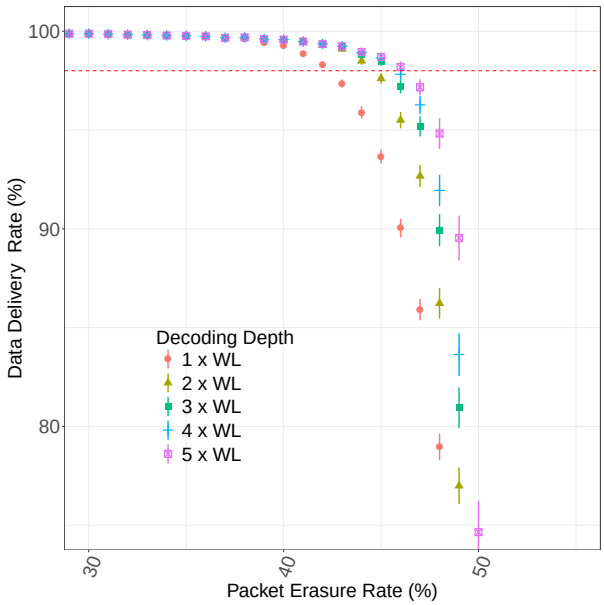

Fig. 7: Simulated DDR as a function of PER with LoRaFFEC $(128; 0.6; \mathrm{DD})$ for various DD.

[4] T. Attia, M. Heusse, B. Tourancheau, and A. Duda. Experimental Characterization of Packet Reception Rate in LoRaWAN. In Rencontres Francophones sur la Conception de Protocoles, l'Évaluation de Performance et l'Expérimentation des Réseaux de Communication, editor, CoRes, Narbonne, France, June 2019. https://hal.archives-ouvertes.fr/CORES2019/hal-02129199v1.

[5] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley. A study of lora: Long range & low power networks for the internet of things. Sensors, 16(9):1466, 2016.

[6] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding: Turbo-codes. In IEEE, editor, International Conference onCommunications (ICC), volume 2, pages 1064–1070, Geneva, 1993.

[7] U. Coutaud and B. Tourancheau. Channel coding for better qos in lora networks. In IEEE, editor, International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pages 1–9, June 2018.

[8] R. Gallager. Low-density parity-check codes. IRE Transactions on information theory, 8(1):21–28, 1962.

[9] LoRa Alliance. What is lora? www.lora-alliance.org/What-Is-LoRa, 2015.

[10] LoRa Alliance Technical Committee Regional Parameters Workgroup. Lorawan 1.1 regional parameters. Technical report, LoRa Alliance, 2018.

[11] P. J. 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, pages 97–108. ACM, 2017.

[12] J. Petäjäjärvi, K. Mikhaylov, M. Hämäläinen, and J. Iinatti. Evaluation of LoRa LPWAN technology for remote health and wellbeing monitoring. In IEEE, editor, International Symposium on Medical Information and Communication Technology (ISMICT), pages 1–5, 2016.

[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 IEEE, editor, International Conference on ITS Telecommunications (ITST), pages 55–59, 2015.

[14] A.-I. Pop, U. Raza, P. Kulkarni, and M. Sooriyabandara. Does bidirectional traffic do more harm than good in LoRaWAN based LPWA networks? In IEEE, editor, Global Communications Conference (GLOBECOM), pages 1–6, 2017.

[15] 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 IEEE, editor, International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), pages 84–89, 2015.

[16] Semtech Corporation. LoRa Modulation Basics. Technical Report AN1200.22, Semtech, 2015.

[17] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE transactions on Information Theory, 13(2):260–269, 1967.