Multi-Gateway Demodulation in LoRa

Alexandre Guitton, Megumi Kaneko

Multi-Gateway Demodulation in LoRa

Alexandre Guitton*†, Megumi Kaneko‡ *IEEE Senior Member*
* Université Clermont-Auvergne, CNRS, Mines de Saint-Étienne, Clermont-Auvergne-INP, LIMOS,
63000 Clermont-Ferrand, France.
† Univ Lyon, INSA Lyon, Inria, CITI, F-69621 Villeurbanne, France.
‡National Institute of Informatics, 101-8430, Tokyo, Japan.
Emails: *alexandre.guitton@uca.fr, †megkaneko@nii.ac.jp

***Abstract*—LoRa is one of the most prominent low power wide area network technologies, and enables to interconnect thousands of devices distributed over areas of several square kilometers. However, the limited number of demodulators present in the hardware of LoRa gateways limits LoRa scalability. In this paper, we argue that scalability can be improved by having gateways collaborate, so that they attempt to demodulate different frames. We propose several algorithms in order to measure the benefits of random-based collaboration and deterministic collaboration. Our simulation results show that random-based protocols improve the baseline performance in most setups, while deterministic protocols improve the network performance when the number of gateways is large, and with many demodulators per gateway.**

I. Introduction

LoRa (Long Range) is a low-rate wide area network technology based on a chirp spread spectrum modulation. It is often used jointly with LoRaWAN, a simple MAC protocol, in which end-devices communicate to a network server through gateways. LoRa and LoRaWAN have been used successfully in various applications [1], thanks to the long communication range of LoRa (i.e., a few kilometers in most scenarios) and to the low cost of a LoRaWAN deployment. The main drawback of these protocols is the low data rate they can achieve.

LoRaWAN gateways are able to demodulate simultaneously several LoRa signals, as long as they are sent on different channels or with different Spreading Factors (SF). To do so, the gateways possess an SX1301 chip [2] which contains eight demodulator chips. Several works have shown that the limited number of demodulators within a gateway limits the performance of the whole network [3], [4], [5]. In [6], we proposed two algorithms that aim to optimize the usage of demodulators in a single gateway environment.

In this paper, our aim is to improve the usage of demodulators in a multi-gateway network in order to improve the number of successfully received frames at the network server level. Indeed, in LoRaWAN, as long as a frame is decoded by one gateway, it is forwarded to the network server. Thus, our goal is to ensure that the gateways collaborate to avoid demodulating the same redundant frames, and instead, to make them demodulate diverse incoming frames, so as to maximize the total number of successfully decoded frames. To do so, our proposed algorithms leverage the inherent properties of LoRa frames, by adapting the gateway collaboration method according to some SF-specific features such as the time-on-air.

Our contributions are the following:

1) We discuss the impact of legacy demodulator allocation algorithms in multi-gateway setups. We show that deploying several gateways at the same location brings only little performance improvement.
2) We propose a deterministic algorithm in order to deal with frames having a large SF, as their large time-on-air enables explicit collaborations among gateways.
3) We propose random selection-based algorithms in order to deal with frames having a small SF, as their small time on air hinder explicit collaborations among gateways.
4) We compare all these algorithms in terms of the number of decoded frames and fairness.

The remainder of this paper is organized as follows. Section II describes the LoRa physical layer (including the usage of demodulators in the hardware architecture of the SX1301 component) and the existing mono-gateway demodulation algorithms. Section III introduces the problem. Section IV first proposes our algorithm based on explicit collaboration through control messages, and then proposes our random selection-based algorithms. Section V details our simulation results in various scenarios. Finally, Section VI concludes our work.

II. LoRa and mono-gateway algorithms

In this section, we first present the LoRa modulation. Then, we present the functioning of the hardware of existing gateways, and the current demodulation algorithms.

A. LoRa modulation

LoRa modulation is based on symbols, also called chirps, which are linear frequency sweeps over a given bandwidth. Up-chirps are symbols whose frequency is increasing over time, while down-chirps are symbols whose frequency is decreasing over time. The duration of a symbol is controlled by the SF, which enables to trade bit rate with robustness and thus with communication range. It is important to note that SFs are quasi-orthogonal: two signals sent simultaneously on different SFs cause only limited interference on each other.

A LoRa frame is composed of a preamble of 12.25 symbols and a payload. The preamble is composed of 8 up-chirps to synchronize the receiver, 2 up-chirps to identify the network, and 2.25 down-chirps to delimit the end of the preamble. The preamble is composed of an optional header and of encoded data. Figure 1 shows an example of

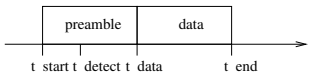


Fig. 1. Times of a frame transmission/reception.

TABLE I
TIME ON AIR AND DECISION TIME FOR VARIOUS PAYLOAD SIZES AND SF.

SF	Payload size			Decision time
	40 bytes	20 bytes	10 bytes	
12	1974.27 ms	1318.91 ms	991.23 ms	270.35 ms
11	1069.06 ms	741.38 ms	577.54 ms	135.14 ms
10	534.53 ms	370.69 ms	288.77 ms	67.57 ms
9	287.74 ms	185.34 ms	144.38 ms	33.83 ms
8	154.11 ms	102.91 ms	72.19 ms	16.91 ms
7	82.18 ms	56.58 ms	41.22 ms	8.42 ms

a frame. In the following, we denote by t_{start} the beginning of the transmission of the preamble of a frame, by t_{detect} the detection time of the preamble, by t_{data} the beginning of the transmission of the payload (which also corresponds to the end of the preamble), and by t_{end} the end of the payload.

Table I shows the time-on-air of frames of various SFs and payload size. The table also shows the time between the detection of the preamble and the beginning of the payload, called decision time. We assumed that it is equal to 4 symbols. This assumption is consistent with other studies: a receiver locking time of 4 symbols is obtained in [7], and 6 symbols are shown to be required for a successful synchronization in [8]. Note that the large decision time (i.e., larger than 50 ms) of large SFs suggests that the packet arbiters of different gateways might be able to communicate while considering which frames to demodulate. Indeed, LoRaWAN gateways are inter-connected through a high-speed IP backhaul.

B. Conventional demodulation algorithms

The SX1301 component is a chip used in the vast majority of LoRaWAN gateways. This chip continuously listens for preambles on all channels and all SFs. When a preamble is detected, the packet arbiter of the SX1301 decides whether one of the eight demodulators should be configured to demodulate the corresponding frame. However, the actual behavior of the packet arbiter of the SX1301 is not documented. We assume that its default behavior follows a simple First-In First-Out (FIFO) strategy: when a new preamble is detected, the first idle demodulator is booked for this frame until t_{end}. If no demodulator is idle, then the frame is ignored.

We showed in [6] that a lightweight recursive approach can significantly improve the performance of a single gateway network, and we proposed two strategies: FIFO-RR1 and FIFO-RR2, whose details are recalled next for clarity. FIFO-RR1 (FIFO with recursive reuse, version 1) reuses booked demodulators during the time between the detection of the preamble and t_{data}, as the demodulators are not processing the payload until t_{data}. Algorithm 1 corresponds to the algorithm used when the preamble is detected, and Algorithm 2 corresponds to the algorithm used after a frame is demodulated. FIFO-RR2 (FIFO with recursive reuse, version 2) enables to plan a future demodulation for a busy demodulator, as long as the new demodulation starts after the end of the current demodulation. Algorithm 3 corresponds to the algorithm used when the preamble is detected, and Algorithm 2 is used after a frame is demodulated.

Algorithm 1 Demodulator allocation for FIFO-RR1.

Require: a new preamble is detected on SF s at t_{detect}
  if there is a demodulator d such that $state[d] = IDLE$ or ($state[d] = BOOKED$ and $next[d] > t_{end}$) **then**
    $next[d] \leftarrow t_{data}$
    push $next[d]$ on $timeStack[d]$
    push the frame parameters on $frameStack[d]$
    $state[d] \leftarrow BOOKED$
    $demodulator[d] \leftarrow$ parameters of the frame
    start a timer for expiration at t_{data}
  else
    frame is ignored
  end if

Algorithm 2 Demodulator reuse after the demodulation, for FIFO-RR1 and FIFO-RR2.

Require: a demodulator d finishes demodulating a frame
  pop $timeStack[d]$ and $frameStack[d]$
  if $timeStack[d]$ is empty **then**
    $state[d] \leftarrow IDLE$
    $demodulator[d] \leftarrow \emptyset$
  else
    $state[d] \leftarrow BOOKED$
    $next[d] \leftarrow$ top of $timeStack[d]$
    $demodulator[d] \leftarrow$top of $frameStack[d]$
    start a timer for expiration at $next[d]$
  end if

Algorithm 3 Demodulator allocation for FIFO-RR2.

Require: a new preamble is detected on SF s at t_{detect}
  execute Algorithm 1
  if the frame was rejected by Algorithm 1 **then**
    foreach demodulator d such that $state[d] = BUSY$ **do**
      $dur \leftarrow$payload duration of top of $frameStack[d]$
      $t_{end}[d] \leftarrow next[d] + dur$
    end foreach
    if there is d such that $state[d] = BUSY$ and $t_{end}[d] \leq t_{data}$ and size of $frameStack[d] = 1$ **then**
      insert t_{data} at 2nd position of $timeStack[d]$
      insert new param. at 2nd position of $frameStack[d]$
    else
      frame is ignored
    end if
  end if

III. PROBLEM FORMULATION

Let $\mathcal{G}$ be a set of gateways. Let $\mathcal{F}$ be a set of frames. Each frame $f \in \mathcal{F}$ is sent by an end-device and is received

by a subset $\mathcal{G}'$ of gateways $\mathcal{G}' \subset \mathcal{G}$. The transmission of the preamble of frame f starts at time t^f_{start}, it is detected by the gateways at time t_{detect} approximately (depending on the propagation time and on the distance between the end-device and each gateway), the transmission of the payload starts at t^f_{data}, and the overall transmission ends at t^f_{end}. For f to be correctly demodulated by a gateway $g \in \mathcal{G}'$, one of the eight demodulators of g has to be continuously allocated to frame f between $[t^f_{data}; t^f_{end}]$ (and not $[t^f_{detect}; t^f_{end}]$). Our goal is to maximize the number of frames that are demodulated by at least one gateway.

Mono-gateway algorithms consider that $\mathcal{G}'$ is a singleton, and thus aim to maximize the number of frames from $\mathcal{F}$ using a centralized scheduling algorithm. In order to show the drawback of mono-gateway algorithms, let us consider a simple scenario where a network operator intends to deploy a second gateway in a dense network, to balance the load between the gateways. The network operator might be inclined to deploy the second gateway at the same location as the first gateway, for simplified logistics. However, this would be a very bad decision if both gateways run a mono-gateway algorithm. Indeed, in this case, both gateways will receive the same frames with similar reception power. The FIFO algorithm will yield the same decisions, so they will both decide to demodulate the same frames, and their demodulator capabilities will be exhausted simultaneously. In other words, the performance of this two-gateway network will be very close to the performance of the previous mono-gateway network, despite consuming twice more demodulating resources.

Multi-gateway algorithms aim to minimize the number of gateways that simultaneously demodulate the same frame, in order to minimize the total demodulator usage. To do this, the gateways can collaborate during $[t^f_{start}; t^f_{data}]$ in order to decide which one should demodulate a frame f. This decision time is however short, as shown in the last column of Table I: it varies between 8.42 ms and 270.35 ms, depending on the SF. If the coordination between gateways is achieved through control messages, this means that the backhaul interconnecting the gateways should have a small latency.

In the remainder of this paper, we study two types of algorithms: a deterministic algorithm which focuses on explicit control messages, thus requiring high-speed backhauls or large SFs, and random algorithms based on differentiation through random decisions, supporting the other configurations.

IV. Proposed algorithms

We start by describing a deterministic algorithm intended for large SFs, and then algorithms intended for small SFs. Then, we raise a discussion on their requirements.

A. Deterministic algorithms for large SFs

Our deterministic algorithm is based on the assumption that gateways can communicate during the short time interval between the preamble detection and the payload start. This time duration depends on the SF, and is given on the last column of Table I. We consider two versions of this algorithm. The *CollaborationPerfect* version assumes that gateways can collaborate instantaneously (that is, in less than 8.42 ms), which enables them to collaborate for all SFs. The *CollaborationImperfect(SF$_{\min}$)* version assumes that gateways can collaborate only for frames using a SF of at least SF$_{\min}$. In both versions, when a gateway detects a new preamble, if it has time to do so depending on the SF$_{\min}$ parameter, it asks to the other gateways whether they plan to demodulate the corresponding frame. If another gateway plans to demodulate the frame, it is ignored by the current gateway. Note that the first gateway to detect the preamble always accepts it.

Notice that it is easy to know the SF from the preamble, as the SF is given as input to the packet arbiter. However, it is challenging to identify individual frames during the preamble. Indeed, frames are typically identified by the sender address, which is contained in the frame header, located relatively deep in the payload (it is in the FHDR field of the MAC header, which follows the PHY header). To cope with this, we use the property given by Approximation 1.

Approximation 1 (Preamble identification): Two preambles p_1 and p_2, detected by two gateways g_1 and g_2 respectively, are assumed to correspond to the same frame transmission if and only if the following four conditions occur: 1) p_1 and p_2 are on the same channel, 2) p_1 and p_2 are on the same SF, 3) the detection times of p_1 and p_2 are within δ_t, and 4) the measured CFO of p_1 and p_2 are within δ_{CFO}.

Note that the two first conditions are necessary. The third condition assumes that the detection time of the two preambles does not differ too much, and requires that the time between gateways is relatively well synchronized. That is why we suggest to use for δ_t a value which depends on the communication time between gateways. The fourth condition uses the fact that each sender has a potential Carrier Frequency Offset (CFO) which yields a shift in the chirp bandwidth. The CFO is detected and corrected during the preamble by LoRa receivers. We did not consider the reception power of p_1 and p_2, as they typically significantly differ for gateways in different locations.

The details of our proposed *CollabPerfect* and *CollabImperfect* are given in Algorithm 4 with parameter SF$_{\min}$ when a preamble is detected. Then, they are based on Algorithm 2 when a demodulation ends, and on Approximation 1 to identify preambles that are already processed by other gateways.

B. Random algorithms for small SFs

Our random algorithms are based on the assumption that gateways do not have enough time for explicit collaboration through control messages. In order to decode as many frames as possible, our goal is to allow gateways to randomly drop planned demodulations when their demodulation capacity is exhausted. In this way, when the traffic density is high, gateways will choose random frames to drop, which will reduce the number of frames that are demodulated simultaneously by several gateways, and

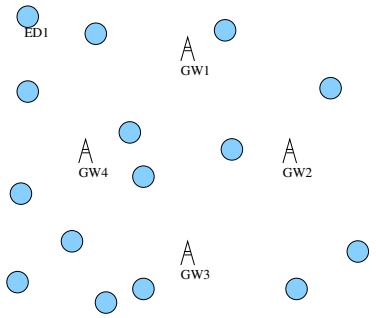


Fig. 2. Example topology for 4 gateways and 15 end-devices (66% are on the left-side, and 34% are on the right-side)

ideally increase the number of frames that are demodulated by at least one gateway.

Our proposed *Random1* algorithm is described in Algorithm 5. Upon detecting a new preamble, it attempts to find an idle demodulator. If none is found, it preempts a random demodulator for the new frame based on a probability P.

Our proposed *Random2* algorithm combines FIFO-RR1 and *Random1* in the following way. (1) In Algorithm 1 of FIFO-RR1, instead of choosing the first demodulator that satisfies a given condition, a random demodulator that satisfies the condition is chosen. (2) If no demodulator is found with Algorithm 1, a preemption occurs with probability P as in *Random1*. When a preemption occurs, a demodulator d is chosen randomly among demodulators having only one planned frame, or among all demodulators otherwise. All the frames planned for d are dropped, and the demodulator is planned for the incoming frame.

C. Discussion

In all our proposed algorithms except *Random1*, as well as in FIFO-RR1 and FIFO-RR2 from [6], we assume that: (1) the maximum duration of a LoRa frame has to be known, and (2) the preamble detection occurs as early as possible.

Concerning the maximum duration of a LoRa frame, most algorithms require to know the time t_{end} where a given demodulator will be available again. Indeed, the length of the frame is not known from the preamble. However, a maximum frame duration exists in LoRa, due to the bounded size of the payload. If the frame happens to be shorter than the expected duration, opportunities of demodulations might be missed, but all algorithms would still operate correctly. Note that it is possible to correct the frame ending time when the demodulator extracts the payload size field from the payload.

Concerning the preamble detection, most algorithms require to know the time where the payload starts. This is usually done by assuming that the preamble is always detected 4 symbols after the beginning of the preamble. If the preamble detection is delayed, the payload can start earlier than expected. In this case, the packet arbiter would detect the start of the payload (thanks to the start of frame delimiter, at the end of the preamble), and would notify the demodulator. The demodulator would have to decide whether finishing the potential current demodulation and missing the new demodulation that is starting, or cancelling the potential current demodulation and starting with the new one. While both decisions reduce the performance of the algorithms, the algorithms still operate. Moreover, we believe that this phenomenon appears rarely in practice. Indeed, the high robustness of LoRa chirps enables to detect preamble chirps quickly even if the channel is noisy. Recall that LoRa is able to decode frames with small SNRs (down to -7.5 dB for SF=7 and to -20 dB for SF=12).

Algorithm 4 Demodulator allocation for *CollaborationPerfect*($\text{SF}_{\min}$=7) and *CollaborationImperf*($\text{SF}_{\min}$ >7)

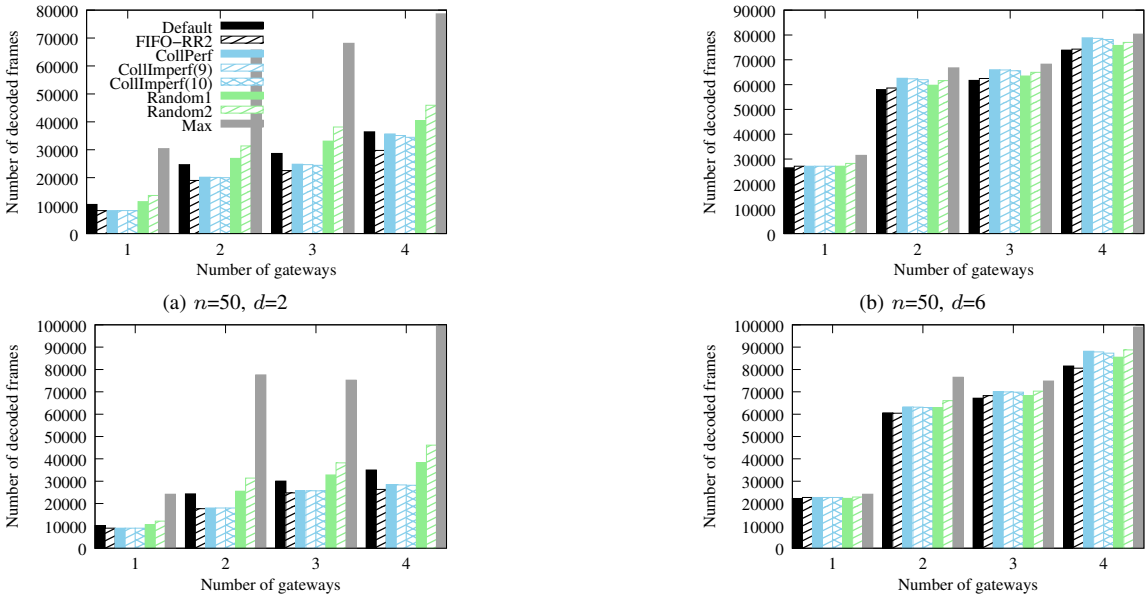
Require: a new preamble p is detected on SF s at t_{detect}
  if there is d such that $state[d] = IDLE$ or ($state[d] = BOOKED$ and $next[d] > t_{end}$) **then**
    if s <$\text{SF}_{\min}$ or no other gateway processes p **then**
      $next[d] \leftarrow t_{data}$
      push $next[d]$ on $timeStack[d]$
      push the frame parameters on $frameStack[d]$
      $state[d] \leftarrow BOOKED$
      $demodulator[d] \leftarrow$ parameters of the frame
      start a timer for expiration at t_{data}
    end if
  else
    foreach demodulator d such that $state[d] = BUSY$ **do**
      $dur \leftarrow$payload duration of top of $frameStack[d]$
      $t_{end}[d] \leftarrow next[d] + dur$
    end foreach
    if there is d such that $state[d] = BUSY$ and $t_{end}[d] \leq t_{data}$ and size of $frameStack[d] = 1$ and (s <$\text{SF}_{\min}$ or no other gateway is processing the frame) **then**
      insert t_{data} at 2nd position of $timeStack[d]$
      insert new param. at 2nd position of $frameStack[d]$
    else
      frame is ignored
    end if
  end if

Algorithm 5 Demodulator allocation for our *Random1(P)* algorithm

Require: a new preamble is detected on SF s at t_{detect}
  if there is d such that $state[d] = IDLE$ **then**
    plan the frame for demodulation with d
  else
    if random number $r \in [0; 1]$ is such that $r \leq P$ **then**
      $d \leftarrow$a random demodulator
      replace the demodulation for d with the new frame
    end if
  end if

V. Simulation results

In order to compare the performance of all these algorithms, we used a simulator developed in Java. The area is a square of 8km×8km. Gateways are deployed in a regular manner: if there is a single gateway, it is located at the center of the area, otherwise, they are deployed regularly



(a) n=50, d=2 (b) n=50, d=6

(c) n=200, d=2 (d) n=200, d=6

Fig. 3. Number of decoded frames for the algorithms, with perfect preamble identification. (a) 50 nodes and 2 demodulators per gateway, (b) 50 nodes and 6 demodulators per gateway, (c) 200 nodes and 2 demodulators per gateway, and (d) 200 nodes and 6 demodulators per gateway.

TABLE II
MAIN SIMULATION PARAMETERS

Area size	8000m×8000m	Duty cycle	50%
Path loss exponent	2.32 (std of 7.8)	Transmit power	14 dBm
Payload size	20 bytes	Capture threshold	6 dB

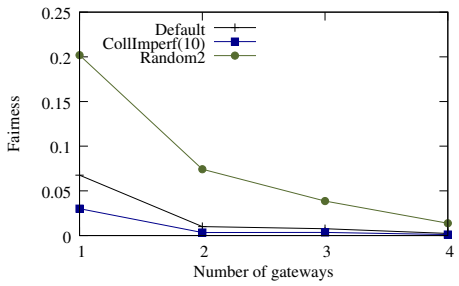
over a circle of 4km-radius, as shown on Figure 2. Nodes are deployed with a hot-spot model: they have a 66% probability to be on the left-side and a 34% probability to be on the right-side. Simulations were also conducted with a homogeneous deployment of nodes, with similar results, but are not shown here due to lack of space. The detailed parameters are given in Table II. The path loss model follows the parameters of [9]. Nodes transmit on the same channel with a duty cycle of 50%, using the smallest SF possible based on the reception sensitivity of the closest gateway. Nodes always have frames to transmit. They are captured if their SINR is larger than 6 dB. Unless stated otherwise, we assume a perfect preamble identification. Each simulation lasts for 1000 s, and each simulation point is averaged over 100 repetitions.

We use two metrics: the number of decoded frames and the fairness. The number of decoded frames is the total number of frames received by at least one gateway. The fairness is defined as $(\sum_{s=7}^{12} p(s))^2/(\sum_{s=7}^{12} p(s)^2)$, where $p(s)$ is the percentage of frames decoded for SF s. Note that we used here the percentage of frames per SF in order to obtain a normalized fairness. Indeed, there might be a large variability in the number of frames for the various SFs in our simulation.

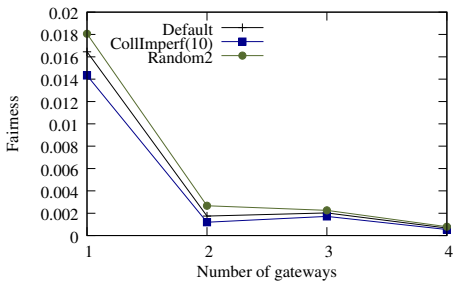
In the following figures, we compare several algorithms. The algorithms in light grey are the existing algorithms: LoRa is in solid, and FIFO-RR2 is in dashed. The algorithms in blue are the proposed deterministic algorithms: *CollPerfect* is in solid, and two *CollImperfect* instances are dashed: one with $\text{SF}_{\min}$=9 and the other with $\text{SF}_{\min}$=10. The algorithms in green are the proposed random algorithms: *Random1* is in solid, and *Random2* is in dashed. Finally, the algorithm in dark grey assumes an infinite number of demodulators per gateway.

Figure 3 shows the number of decoded frames with perfect preamble identification, as a function of the number of gateways, for n nodes and d demodulators per gateway, with (a) $n = 50$ and $d = 2$, (b) $n = 50$ and $d = 6$, (c) $n = 200$ and $d = 2$, and (d) $n = 200$ and $d = 6$. It can be seen that FIFO-RR2 improves LoRa only when the number of demodulators is large. The performance of proposed *CollPerf* and *CollImperf* method improves as the number of gateways increases, which is an expected behavior for such collaborative algorithms. Their performance exceeds that of LoRa when the number of demodulators is large. Moreover, it can be seen that the performance degradation between *CollPerf* and *CollImperf(10)* is small, which makes *CollImperf(10)* a good candidate in practice. The performance of random algorithms is always better than LoRa: their performance is typically very high when the number of demodulators is low, and decreases with the number of demodulators.

Figure 4 shows the fairness with perfect preamble identification, as a function of the number of gateways, for 200 nodes and d demodulators per gateway,with (a) $d = 2$, and (b) $d = 6$. Only three algorithms are displayed: the Default algorithm, *CollImperf(10)* (which is the collaborative algorithm with the most realistic parameter), and *Random2* (which is the random algorithm with the best performance). The fairness reduces with the number of gateways, for all algorithms. Indeed, when the number of gateways is large, most nodes are close to at least one gateway, and therefore most nodes use a small SF. As the algorithms accept a large number of short frames (of SF7), the relative impact of the ignored frames of SF12 increases. This shows that it is difficult to increase the throughput while increasing the



(a) $n = 200$, d=2



(b) $n = 200$, d=6

Fig. 4. Fairness for three algorithms, with perfect preamble identification and 200 nodes. (a) 2 demodulators per gateway, and (b) 6 demodulators per gateway.

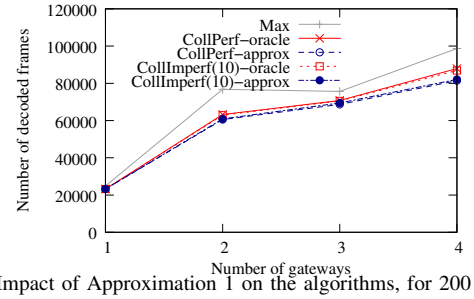


Fig. 5. Impact of Approximation 1 on the algorithms, for 200 nodes and 6 demodulators per gateway.

fairness, as throughput is typically increased by accepting more short frames, while fairness is typically increased by accepting more long frames (since they are less frequent). *Random2* has the best fairness, especially when the number of demodulators is low.

Next, we evaluate the impact of imperfect preamble identification. In order to do so, we implemented in each node a random CFO within $[-25\%; 25\%]$, and we set $\delta_{CFO} = 5\%$. We assumed that the detection time of a preamble is within $[4t_s; 5t_s]$, where t_s is the symbol duration, and we set $\delta_t = t_s/4$. Then, we compared the performance of *CollPerfect* and *CollImperfect(10)* without a perfect preamble identification, and with this imperfect implementation of Approximation 1. We run the simulation for 200 nodes and 6 demodulators per gateway. Note that with the imperfect identification, it is possible for two gateways to incorrectly assume that two preambles correspond to different frames although they correspond to the same frame (false negative), or to incorrectly assume that two preambles correspond to the same frame although they do not (false positive). In the former case, they might both decode the frame, which reduces the number of decoded frames. In the latter case, a gateway might ignore the new frame as it believes the other gateway is already demodulating it, which also reduces the number of decoded frames.

Figure 5 shows the impact of Approximation 1 on the number of decoded frames by *CollPerfect* and *CollImperfect(10)*, as a function of the number of gateways, for 200 nodes and 6 demodulators per gateway. For both algorithms, there is a small gap between the performance of the algorithm with the oracle (i.e., with perfect identification), and the performance of the more realistic setup without the approximation. Although this gap increases with the number of gateways, up to 7.38% for *CollPerfect* and 5.53% for *CollImperfect(10)*, it remains very limited, thereby validating the proposed approaches under practical conditions.

VI. Conclusions

The limited number of demodulators in LoRa gateways is known to impact the performance of single-gateway networks. In this paper, we have proposed the first multi-gateway demodulator allocation algorithms. These algorithms aim at improving the scalability by reducing the number of gateways that demodulate the same frames. A deterministic algorithm is tailored to the case where gateways have enough time to exchange preamble information through control messages, while random algorithms focus on the case where the SF is so small that gateways have to make decisions without explicit collaboration. Our simulation results show that all the proposed algorithms can improve the performance of LoRa: deterministic algorithms show better performance when the number of gateways and demodulators is high, while random algorithms always have good performance, but shine when the number of demodulators is small and the number of gateways is large. Based on these initial findings, we will extend the proposed methods by designing integrated approaches of random selection and distributed optimization.

References

[1] L. Kolobe, B. Sigweni, and C. K. Lebekwe, "Systematic literature survey: applications of lora communications," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 3, pp. 3176–3183, 2020.

[2] Semtech, "SX1301 - digital baseband chip LoRaWAN macro gateways," Wireless & Sensing Products, datasheet version 2.4, 06 2017.

[3] R. B. Sorensen, N. Razmi, J. J. Nielsen, and P. Popovski, "Analysis of LoRaWAN uplink with multiple demodulating paths and capture effect," in *IEEE ICC (International Conference on Communications*, 2019.

[4] P. K. Dalela, S. Sachdev, and V. Tyagi, "LoRaWAN network capacity for practical network planning in india," in *URSI AP-RASC (Asia-Pacific Radio Science Conference)*, 2019.

[5] D. Magrin, M. Capuzzo, and A. Zanella, "A thorough study of LoRaWAN performance under different parameter settings," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 116–127, 01 2020.

[6] A. Guitton and M. Kaneko, "Improving LoRa scalability by a recursive reuse of demodulators," in *IEEE Globecom*, 2020.

[7] A. Rahmadhani and F. Kuipers, "When LoRaWAN frames collide," in *WiNTECH (International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization)*, 2018.

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

[9] J. Petajajarvi, M. Pettissalo, K. Mikhaylov, T. Hänninen, and A. Roivainen, "On the coverage of LPWANs: Range evaluation and channel attenuation model for LoRa technology," in *ITST (International Conference on ITS Telecommunications)*, 12 2015.