Autonomous Collision-Free Scheduling for LoRa-based Industrial Internet of Things

Dimitrios Zorbas and Brendan O'Flynn
Tyndall National Institute, University College Cork, Ireland
Email: dimzorbas@ieee.org

***Abstract*—LoRa-based transmissions suffer from extensive collisions even for low node numbers due to unregulated access to the medium. In order to tackle this problem, we propose a collision-free time-slotted scheduling approach where each node autonomously decides when to transmit a packet based on its unique identifier which is converted to a slot number using a modulo operation. We report through simulations and real experiments that this approach can provide very high reliability when the nodes are synchronized. Moreover, it does not require any additional communication overhead apart from a broadcast packet emitted by the gateway. Our comparison with the native LoRa, as well as to a slotted-LoRa version, shows significant performance gains in terms of packet delivery ratio, especially in the case of low node populations.**

I. Introduction & Background

LoRa is a proprietary physical layer communication protocol operating in the sub-GHz license-free band and it is designed for low-power, long range, and low bandwidth IoT applications. LoRa trades data rate with signal sensitivity. To achieve this, it uses a spread spectrum modulation to select the amount of spread used for a fixed bandwidth. This spread is controlled using a radio parameter, called Spreading Factor (SF). In practice, this means that long range transmissions are performed with a lower data rate (higher SF) than in short range transmissions (lower SF).

The LoRa Alliance has proposed an open upper layer stack to manage communications between LoRa gateways and end-node devices, called LoRaWAN. LoRaWAN is responsible for managing a secure joining procedure by exchanging network keys. It is also responsible for managing communication parameters like the data rate and the transmission power of end devices, through the Adaptive Data Rate (ADR) mechanism [1].

The main issue of a LoRaWAN network is its scalability. The transmissions are treated as ALOHA transmissions, which leads to a high probability of collisions and, thus, to a poor performance even for low node numbers [2], [3]. Moreover, the frequency of LoRa transmissions is regulated by either an upper duty cycle limit (i.e., 1% for uplink transmissions in the EU868 band) or by a maximum transmission time per channel (US regulations for sub-GHz bands) [4] which can decrease the performance, especially in presence of confirmable applications [5]. Finally, it has been shown that the capture effect and the inter-SF interference can further decrease the network capacity [6], [7].

A number of solution have been presented in the literature in order to tackle the burst of collisions of LoRa(WAN), mainly through the scheduling of the node activities. A lightweight scheduling approach is proposed by Reynders *et al.* [8]. The nodes are divided into groups with similar transmission powers in order to reduce the capture effect. The schedule moves some nodes to higher SFs in order to allow collision-free simultaneous transmissions. However, this solution requires significant communication overhead and also increased energy consumption for some of the nodes. An on-demand time division protocol to assign slots to the nodes has been proposed by Haxhibeqiri *et al.* [9]. The solution improves the network performance but does not completely eliminate collisions. Abdelfadeel *et al.* [10] present a two-phase collision-free method for fast bulk data delivery using the mobile gateway concept [11]. The work schedules spreading factors and assigns transmission powers, frequency channels, and timeslots to LoRa(WAN) end-devices. Significant performance improvements are reported but with additional and significant overhead. Finally, a slotted ALOHA overlay on LoRaWAN is proposed by Polonelli *et al.* [12], where the nodes are synchronized according to the gateway's clock. Similarly to slotted-ALOHA, this approach alleviates the number of collisions but does not eliminate them.

Unlike the current works in the literature, in this paper, we propose a collision-free time-slotted scheduling approach which exhibits an almost negligible overhead. We take advantage of the unique – already registered – device IDs, and we convert the corresponding MAC addressees to unique integers and, then, to slot numbers using a modulo operation. This approach does not guarantee optimal schedules in terms of length, since it leaves empty slots in between transmissions. The number of empty slots tends to increase with the number of nodes in the network, resulting in long frame lengths. However, even in this case, we show through simulations and experiments that it exhibits an almost 100% packet delivery ratio and faster data collection compared to the native LoRa and a random slotted-LoRa approach.

II. LoRa-based Autonomous Scheduling

In this section, we describe how a specific number of LoRa end-devices can autonomously compute the number of their transmitting slot given their unique identifier and a broadcast packet they get from the gateway. The described process works for devices from the same manufacturer, however, as

Algorithm 1: Gateway: MAC to int conversion

require: MAC (64bits MAC address in hex format)
$MAC = \texttt{substr}(MAC,\ 9)$; `// subtract the first 9 characters`
$mac\_int = \texttt{int}(MAC,\ 16)$; `// convert hex to int`
return mac_int;

Algorithm 2: Node: MAC to slot conversion

require: MAC (64bits MAC address in hex format), k
$MAC = \texttt{substr}(MAC,\ 9)$; `// subtract the first 9 characters`
$mac\_int = \texttt{int}(MAC,\ 16)$; `// convert hex to int`
$slot = mac\_int \bmod k$;
return $slot$;

we explain later in the text, it can be easily converted to a global solution.

A. MAC to slot conversion

LoRa devices have a 64bit unique identifier (DevEUI) that is assigned to the devices by the chip manufacturer. The first 36bits are reserved to identify the manufacturer and the last 28bits are dedicated to the devices. A LoRa gateway (or the backbone network behind the gateway) is aware of the nodes' DevEUIs since they are used during the join request process (for example during OTAA in LoRaWAN). Having this information as well as the number of devices per SF, the gateway can compute a minimum transmission period (frame length) during which all the nodes with the same SF can transmit one packet. The nodes can transmit multiple packets by utilizing multiple frames. The structure of the frame is depicted in Fig. 1. This example shows two frames, one for the SF7 and one for the SF8, consisting of 4 and 2 slots respectively. A guard time has been added in the end of each slot to tolerate slight desynchronizations. We, also, assume that the packet size is the same for all the SFs, thus, the higher the SF, the longer the slot size.

The gateway computes the frame length in a two step process. First, it converts all the MAC addresses to unique 28bits unsigned integers by initially subtracting the first 9 characters (36bits) of the corresponding hex identifiers as it is depicted in Algorithm 1. After doing this for each available MAC address, it generates a set A of n 28bits integers, where n is the number of nodes having the same SF. The next step is to compute the number of slots in the frame based on a modulo operation for all the numbers in A. To do so, the following optimization problem must be solved.

$$\min k, \tag{1}$$

$s.t.$

$$A = \{\alpha_1, \alpha_2, ..., \alpha_n\},\ \alpha_i \neq \alpha_j, \forall\ i, j \in [1, n], \tag{2}$$

$$\alpha_i \bmod k \neq \alpha_j \bmod k, \forall\ i, j \in [1, n]. \tag{3}$$

The input of the problem is a set of n positive integers $A = \{\alpha_1, \alpha_2, ..., \alpha_n\}$, where $\alpha_i \neq \alpha_j$, $\forall\ i, j \in [1, n]$. A minimum number of slots k, $k \in \mathcal{Z}$, has to be computed so that Constraint (3) is satisfied.

Once the aforementioned problem has been solved, the number of slots k or the corresponding frame length is communicated to the nodes. The number of slots can be converted to a frame length (and vice versa) using Eq. (4), where $airtime(SF, BW, PL)$ is the transmission time for the given SF, channel bandwidth (BW), and payload size (PL) [1]. The gateway can include the slot numbers of multiple SFs decoded in a single packet using the highest SF in the system so that all the nodes can receive it. If the gateway decides to transmit multiple packets (one per SF), it has to deal with duty cycle limitations (e.g., 1% or 10% in EU) that could add additional delay in the initialization of the network. In any case, the nodes need to be already synchronized when they receive the frame length data. Even though the node synchronization problem is not examined in this paper, it can be easily tackled by periodically sending clock correction information at specific timeslots [13].

$$frame\_length = k \cdot (airtime(SF, BW, PL) + guard). \tag{4}$$

Each individual node uses the information it gets from the gateway as well as its MAC address to compute its transmitting slot. To do so, it uses the MAC-to-slot conversion algorithm as it is presented in Algorithm 2. This is the same algorithm that the gateway uses to convert MACs to integers with an additional modulo operation.

An example of the above procedure is given as follows. Let us assume a set of 5 nodes using SF7 with the following MAC addresses in HEX format: `70b3d5499d64b925`, `70b3d54994053846`, `70b3d549959660b3`, `70b3d549943d50d1`, `70b3d5499fae2761`. The first step of the gateway is to convert these MACs to 28bit integers using Algorithm 1, so the set $A = \{224704805, 67450950, 93741235, 71127249, 263071585\}$ is generated. In the next step, the optimization problem described in Eq. (1) needs to be solved. In this particular case, it is easy to find (by using a number of successive tries starting from $k = 5$) that 9 slots is the optimum solution. On the node side, Algorithm 2 will generate a unique slot number for each individual node. The slot numbers that are generated and allow collision-free transmissions are 5, 0, 7, 6, and 1. Apparently, this solution leaves some empty slots, however, the purpose of this approach is to achieve simplicity and reduced overhead rather than optimality. Moreover, the empty slots can be considered as a positive feature, assuming an application where extra nodes join the network and occupy unused slots.

B. Manufacturer-independent scheduling

The solution described above can be easily extended to a manufacturer-independent solution by adding a few more steps in Algorithms 1 and 2. For this reason, we present Algorithm 3 as a replacement of Algorithm 1. More specifically, each MAC address is hashed with a function like the MD5. Since

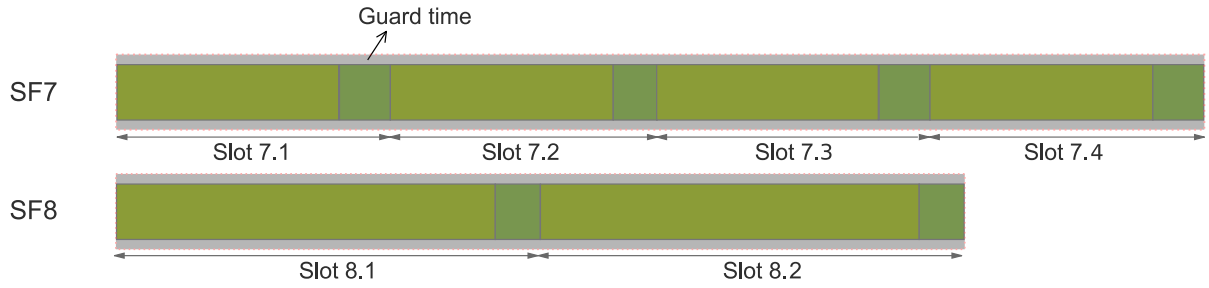


Fig. 1. An example of the frame structure for SF7 and SF8.

Algorithm 3: Gateway: Hardware-independent MAC to int conversion

require: MAC (64bits MAC address in hex format)
$MAC = \texttt{hash}(MAC)$;
$MAC = \texttt{32msb}(MAC)$;
$mac\_int = \text{int}(MAC,\ 16)$; `// convert hex to int`
return mac_int;

the result of the function may be a number of 128 or more bits, we first need to convert it to a 32bit number so it can be practically used in most of the current embedded systems. A solution to do this, is to keep the 32 most significant bits of the hash digest. The integer representation of the number can, then, be used in the modulo operation. We must note here, that the conversion from 128 bits to 32 bits may produce duplicate numbers (and thus slots), so this solution does not guarantee collision-free transmissions. However, the probability of having identical all the 32 first bits of two hashed numbers is extremely small[1]. Moreover, the hash computation adds some additional computation cost for the nodes, however, it can be done fast in modern hardware platforms. We leave this computational analysis as part of our future work.

C. Duty cycle restrictions

Due to the duty cycle restrictions, each particular node can transmit again at least after 99 times the duration of the last transmission (assuming a duty cycle of 1%). This restriction sets a lower limit to the frame length which is related to the packet size and the parameters of the transmission (SF, bandwidth etc.). For example, a node transmitting in the first slot of a frame, where each slot has a length of 30ms (including a 5ms guard time), is allowed to transmit again after $99 \cdot 25$ms. This implies a minimum frame length of $\lceil \frac{99 \cdot 25 + 25}{30} \rceil = 84$ slots and, by extension, all node populations from 1 to 84 nodes will have to obey this minimum frame length. In general, the minimum number of slots for a given transmission time $airtime(SF, BW, PL)$ is computed as follows:

$$slots = \lceil \frac{100 \cdot airtime(SF, BW, PL)}{airtime(SF, BW, PL) + guard} \rceil. \quad (5)$$

[1]It equals to $\frac{1}{16^{32}}$ assuming that all 16 symbols have the same probability of appearance.

D. Performance improvements

In most of the cases, since the scheduler generates frames with several empty slots, a simple approach to remove some of these slots and, thus, reduce the frame length, is to shift the occupied slots towards the first slot and eliminate some of the empty ones. This can be done using a simple *shift* and *elimination* approach as depicted in the example of Fig. 2. In this example, we assume a network consisting of 4 nodes and the scheduler has initially computed a frame length of 11 slots ($k = 11$). The slots with bold numbers are occupied slots and the grey ones are empty slots. During the *shift* operation, all the occupied slots are shifted towards the first slot (slot 0). The number of shifted slots depends on the "distance" of the minimum occupied slot to slot 0. In this example, all the occupied slots are left-shifted by 2 slots. We must note that if, initially in the schedule, slot 0 is already occupied, then the shift operation can be skipped since it has no effect. During the *elimination* step, the minimum empty slots in-between successive occupied slots is computed. In the example, this number equals 1. Thus, all the occupied slots except slot 0 are further left-shifted by 1 slot, so the total number of freed slots is 3. Thus, 3 empty slots in the end of the frame are eliminated. This results to a shorter frame with less empty slots, however it is not applicable in all the cases since it depends on the sparseness of the occupied slots. For example, the second shift operation would have no effect if at least two nodes are assigned to successive slots.

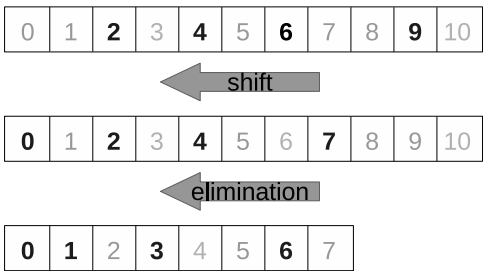


Fig. 2. An example of the shift and elimination improvement.

In terms of communication overhead, the gateway has to include the total number of shifted slots into the broadcast packet apart from the initial frame length. This will require 4 extra bytes.

TABLE I
EXPERIMENT PARAMETERS

Parameter	Value
Nodes	10 + 1
Bandwidth (BW) - Coding Rate	500 KHz - 4/5
Spreading Factor (SF)	7
Region	EU868
Preamble symbols	8
Guard time	5ms
Packet size	50 Bytes
Transmissions per node	40
Transmission power	2 dBm
Execution times	20

III. PERFORMANCE EVALUATION & DISCUSSION OF THE RESULTS

In this section, we assess the performance of the proposed approach using a set of simulations and experiments. The purpose of the simulations is to test the slot generation process using a high number of nodes and repeats, something that could not be possible – at least in a reasonable amount of time – using real experiments. Finally, we perform some experiments to test our approach on real devices.

A. Setup

We compare our approach to the native LoRa, where nodes choose a random time to send a packet, as well as to a slotted-LoRa version, where the nodes are synchronized and the transmissions are performed at random slots. A LoRa simulator using a proper capture effect and a path-loss model was used to conduct the simulations[2]. We, also, optimally solve the optimization problem of Eq. (1) using exhaustive search starting from $k = n$ with an increment of 1. We consider 40 packet transmissions per node for all the approaches. In order to get results faster, we assume that all the nodes use SF7, however, a higher SF would only change the length of the slot and the total simulation or experiment time. Every instance of the simulation is executed 50 times using random node topologies and the average results are presented along with the 95% confidence intervals. For the needs of the experiments, a testbed consisting of 10 nodes and a gateway is used, all operating at the same frequency, channel bandwidth, and SF. We conduct experiments considering high traffic scenarios with a total of 400 (4x10) transmissions in a time window of about 34 seconds (as a result of the optimal frame length). The experiment parameters are summarized in Table I. We must note that our network co-existed with other LoRa(WAN) networks in the building. We have not taken into account the improvement of Section II-D into the results.

B. Results

Fig. 3 presents the simulation results using different node populations with up to 300 nodes. The results of figure (a) reveal that when the number of nodes is low all the transmissions can be performed without collisions in a short time, however, this time increases exponentially and may not scale well with hundreds of nodes. This is because of the difficulty of the solver to find modulo divisors that satisfy Constraint (3) as n is getting higher. This is also explained by figure (b) as the execution time increases with the number of nodes, even though more time efficient solutions can be developed using, for example, linear programming. Nevertheless, the results are very positive for lower number of nodes (e.g., 10-50) since the frame size is kept low, while the execution time is negligible.

Fig. 3c reports the packet collision rate when using the Slotted-LoRa approach (Random slotted) and the native LoRa (Default LoRa). We set the frame size equal to the one computed by the MAC-based approach (as depicted in Fig. 3a). This actually affects the frequency of the transmissions for the native LoRa since every node randomly uses a time within the frame bounds to transmit a packet. The results reveal that for the same amount of time the two compared approaches achieve very high collision rates that vary from 9 to 48% for native LoRa, and 4 to 35% for slotted-LoRa. Apparently, as we move to higher node populations, the gap with the MAC-based scheduling is decreasing. As expained in the previous paragraph, this is because the number of empty slots increases a lot, increasing the frame size as well.

Finally, we test the performance of the MAC-based approach using the 10-node testbed. Given the specific MAC addresses, the optimal solution is 28 slots (or 840ms) for SF7. This results to a total experiment time of 33.6 seconds for all 40 transmissions per node. The results are illustrated in Fig. 4 and confirm the simulation results. In particular, they show a Packet Delivery Ratio (PDR) close to 1, while the default LoRa PDR barely exceeds 0.6 within the same data collection window. We repeated the experiment multiple times with different software-generated MAC addresses and the results were similar. Finally, we experimentally tested different values of guard time and we observed that it can be reduced to ¡1ms if the nodes are perfectly synchronized. Shrinking the guard time can considerably decrease the frame length, especially for high node number deployments. However, we need to further investigate this low guard time in long-range deployments.

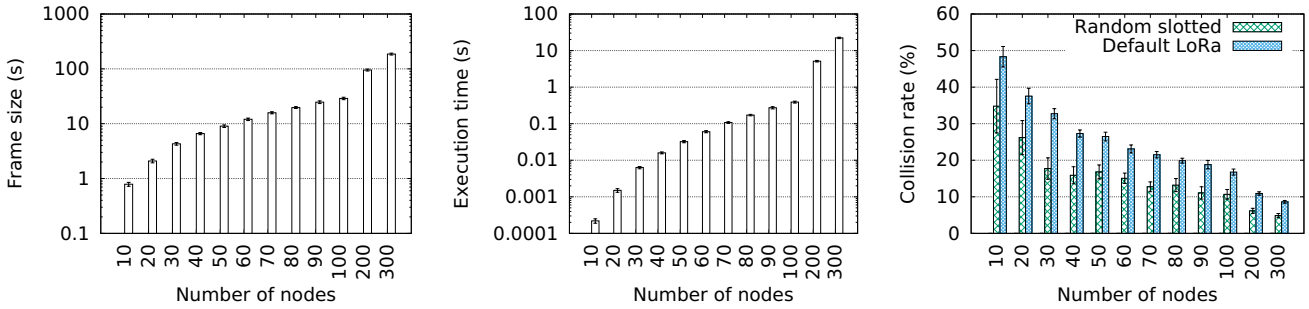
IV. CONCLUSIONS & FUTURE WORK

In this paper, we proposed a collision-free scheduling approach for time-slotted LoRa transmissions. The novelty of the approach is that it uses the MAC addresses of the nodes as an autonomous way to compute the slot of the transmission, while receiving minimal information from the gateway. We showed through simulations and experiments that this solution performs well, at least with a low number of nodes. In the future, we are planning to evaluate the hardware-independent solution in terms of reliability and energy consumption, and also to assess its performance using multiple spreading factors scenarios.

ACKNOWLEDGEMENTS

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI)

[2]https://github.com/deltazita/Bulk-LoRa



(a) Frame size using the MAC-based approach (b) Execution time of the exhaustive search (c) Collision rate of the approaches using the frame size depicted in (a)

Fig. 3. Performance of the MAC-based approach with a collision-free schedule and comparison with other approaches.

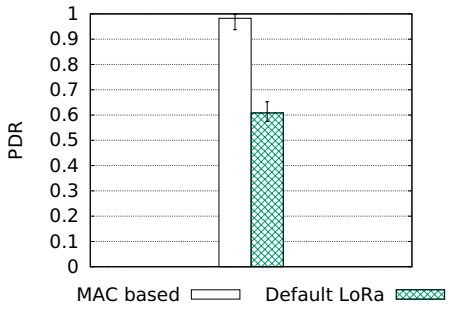


Fig. 4. Packet delivery ratio with 10 nodes and SF7. (The error bars of MAC-based refer to the minimum and the maximum captured values.)

and is co-funded under the European Regional Development Fund under Grant Number 13/RC/2077.

It has also received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 713567.

REFERENCES

[1] LoRa Alliance, "Lorawan 1.1 specification," https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf, 2017.

[2] M. C. Bor, U. Roedig, T. Voigt, and J. M. Alonso, "Do lora low-power wide-area networks scale?" in *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWiM '16. ACM, 2016, pp. 59–67.

[3] D. Bankov, E. Khorov, and A. Lyakhov, "Mathematical model of lorawan channel access," in *WoWMoM*, June 2017, pp. 1–3.

[4] D. Castells-Rufas, A. Galin-Pons, and J. Carrabina, "The regulation of unlicensed sub-ghz bands: Are stronger restrictions required for lpwan-based iot success?" 2018, arXiv:1812.00031.

[5] 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 Global Communications Conference (GLOBECOM)*. IEEE, 2017, pp. 1–6.

[6] D. Croce, M. Gucciardo, S. Mangione, G. Santaromita, and I. Tinnirello, "Impact of lora imperfect orthogonality: Analysis of link-level performance," *IEEE Communications Letters*, vol. 22, no. 4, pp. 796–799, April 2018.

[7] A. Mahmood, E. Sisinni, L. Guntupalli, R. Rondon, S. A. Hassan, and M. Gidlund, "Scalability analysis of a lora network under imperfect orthogonality," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2018.

[8] B. Reynders, Q. Wang, P. Tuset-Peiro, X. Vilajosana, and S. Pollin, "Improving reliability and scalability of lorawans through lightweight scheduling," *IEEE IoT Journal*, vol. 5, no. 3, June 2018.

[9] J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Low overhead scheduling of lora transmissions for improved scalability," *IEEE Internet of Things Journal*, pp. 1–1, 2018.

[10] K. Q. Abdelfadeel, D. Zorbas, V. Cionca, B. O'Flynn, and D. Pesch, "Free - fine-grained scheduling for reliable and energy efficient data collection in lorawan," 2018, arXiv:1812.05744.

[11] D. Zorbas and B. O'Flynn, "Collision-free sensor data collection using lorawan and drones," in *Global Information Infrastructure and Networking Symposium (GIIS)*. IEEE, Oct 2018, pp. 1–5.

[12] T. Polonelli, D. Brunelli, A. Marzocchi, and L. Benini, "Slotted aloha on lorawan-design, analysis, and deployment," *Sensors*, vol. 19, no. 4, 2019.

[13] L. Tessaro, C. Raffaldi, M. Rossi, and D. Brunelli, "Lightweight synchronization algorithm with self-calibration for industrial lora sensor networks," in *2018 Workshop on Metrology for Industry 4.0 and IoT*. IEEE, 2018, pp. 259–263.