Offline Scheduling Algorithms for Time-Slotted LoRa-based Bulk Data Transmission

Dimitrios Zorbas*, Khaled Q. Abdelfadeel†§, Victor Cionca†, Dirk Pesch†§, and Brendan O'Flynn*
*Tyndall National Institute, University College Cork, Ireland
†Nimbus Research Centre, Cork Institute of Technology, Ireland
§Computer Science Dept., University College Cork, Ireland
Corresponding Email: dimitrios.zorbas@tyndall.ie

***Abstract*—LoRa-based transmissions suffer from extensive collisions due to the ALOHA-style transmission policy. As a consequence, delivering a high number of packets in a short amount of time becomes an unfeasible task. To tackle this problem we propose to schedule node transmissions in slots of different size depending on the Spreading Factor (SF). Transmissions with the same SF are scheduled in different slots to avoid collisions while those with different SF can occur in parallel. In this paper, we propose two algorithms executed in an offline manner to allocate SFs to nodes aimed at minimising the total data collection time while respecting the radio duty cycle restriction. The examined simulation scenarios and our comparison with LoRaWAN show an up to 101% improvement in terms of data collection time and an up to 250% improvement in terms of energy consumption combined with a nearly 100% packet delivery ratio.**

I. Introduction

LoRa networks are gaining traction in agricultural and environmental applications [1]. However, an issue that has not been addressed is that rural areas often experience poor Internet connectivity, which creates connectivity problems and the ability to extract data from LoRa gateways. One solution would be satellite or cellular communications, however this solution is expensive. The alternative that we explore in this paper is the use of temporary gateway deployments, where a gateway is brought into the network at regular intervals to collect in bulk all the data gathered and saved by the nodes.

During bulk data collection, nodes aim to transmit continuously which may saturate the channel. However, traffic is predictable and, therefore, suitable for scheduling, which should prevent multiple access collisions. As a result, increasing the throughput and decreasing collection time as well as node energy consumption compared to the traditional asynchronous communication. The goal is to schedule the transmissions for all the nodes. We recently proposed an *online* scheduler with partial network knowledge and showed significant improvements in time and energy consumption [2]. In this paper, we investigate the gains from using *offline* algorithms with complete network knowledge.

The problem of scheduling communication is not new, but in LoRa it represents a particular case that has not been explored (for bulk collection). Bulk collection in traditional sensor networks has had to deal with the complexity of multi-hop communication [3]. LoRa networks do not have this issue due to their long-range single-hop nature, however they are constrained by communication duty cycle limitations [4] and complicated by the use of multiple, pseudo-orthogonal [5] spreading factors. We propose two offline algorithms that both take as input the entire list of nodes already registered in the network, and allocate SFs and slots for transmissions. We propose *Global* algorithm computes the schedule for all transmissions and, as a consequence, has a single frame. The second algorithm, called *Light*, schedules only the first transmission for each node and repeats that in subsequent frames. Compared with LoRaWAN Class-A bulk data collection we obtain substantial improvements in both time and energy consumption. We also discuss the problem of transmitting the computed schedule from the gateway to the nodes, which is constrained by the duty cycle regulations at the gateway. We propose two solutions, compute their overhead and discuss their implications.

II. Related work

It has been shown that even unconfirmable LoRaWAN networks cannot scale due to the ALOHA-based medium access of the LoRaWAN MAC layer which causes a dramatic increase in collisions even for moderate number of transmissions [6], [7]. For this reason, a number of solutions including transmission scheduling have been proposed in the literature to improve network performance.

Scheduling of LoRa-based transmissions has attracted much attention over the past year. Reynders *et al.* [8] propose a two-step lightweight scheduling approach to divide nodes into groups where similar transmission powers are used in each group to reduce the capture effect. The nodes are recommended by the gateway's scheduling to use different SFs to enable simultaneous transmissions and thus reduce packet collisions.

An on-demand scheduling is presented in [9], where nodes request time slots from the gateway. The work proposes to have the gateway send the time slot indices encoded in bloom filters; while this achieves a reduction in size to mitigate the duty cycle limitations, bloom filters are a probabilistic data structure that can have false positives so it does not eliminate collisions. The authors report 7 and 30% increase in packet delivery ratio for SF7 and SF12, compared to default class A.

The potential of using LoRa technology for industrial applications is examined in [10]. The authors compute the theoretical maximum network capacity when synchronisation is assumed and all channels as well as SFs are used by the nodes. However, no specific algorithm is proposed to allocate slots and assign SFs.

Finally, a SF allocation algorithm for fast bulk data collection by LoRa gateways carried by drones is proposed in [11]. The transmissions are scheduled one after the other and significantly reduce the data collection time. However, the 1% radio duty cycle restriction is not taken into account.

Unlike previous works, in this paper we exploit the capability of re-assigning SFs and schedule transmissions in time slots for applications that require fast and reliable bulk data transmissions.

III. Problem considerations

A. Slots and scheduling

In this work, we assume that the time is divided into time slots. Each slot must be long enough to accommodate one transmission which implies variable slot sizes based on the respective packet size and transmission time per SF. The scheduled *frame* is a 2D array with six rows, one per SF, and each consisting of a list of slots. The role of a scheduling algorithm is to compute this frame by finding the best allocation per node or transmission so that the total data collection time is minimised. To do so, transmissions with different SFs can occur in parallel while those with the same SF are accommodated one after the other, eliminating the probability of collisions.

Assuming that nodes have more than one packet to send, transmissions from the same node have a minimum period limited by the duty cycle regulations. The time slots in between can therefore be used to schedule transmissions from other nodes. In general, if the airtime for a SF is α_f and the duty cycle is 1%, the minimum period per node is $\alpha_f \cdot 100$ and another $\eta_f = \frac{99\alpha_f}{\alpha_f + 2\tau}$ nodes can have their transmissions scheduled in the meantime, where τ is the guard time in a slot. If more than η_f nodes are allocated to SF f, the schedule for f is extended. This has the disadvantage of reducing the transmission rate. The schedule is extended until its length *in time* is more than the length of the next $(f+1)$ schedule (considering its minimum transmission period). At that point nodes that have a minimum SF f will be scheduled to transmit in the $f+1$ schedule. In general, a transmission is allocated on the SF that leads to the earliest completion of the data collection for that node. Fig. 1 illustrates the allocation.

B. Synchronisation

As with every time-based scheduling algorithm, the nodes are periodically synchronised by the gateway according to a global clock. Since the clock of the nodes may drift over time, guard times are added between successive transmissions to ensure that the current transmission will not overlap with the next one. The used guard time depends on how often the nodes are synchronised with the global (gateway based) clock as well as on some physical characteristics of the device crystals.

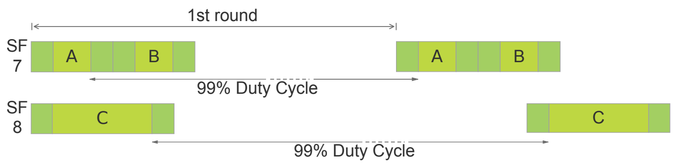


Fig. 1. Slot and SF allocation of three transmissions with 100 bytes packet size and initial SF equal to 7.

C. Packet size

The packet size plays an important role for the schedule length as well as for the network reliability [2]. A smaller packet size decreases the probability of collisions in presence of external interference since the transmission time is lower, but it increases the overhead resulting from the accumulated guard times and MAC headers. The opposite holds true for large packet sizes. As a consequence, finding a trade-off between reliability and overhead is a critical issue when designing scheduling algorithms. However, due to space restrictions we assess only the effect of the packet size on the schedule length and not on the network reliability in presence of external interference. Thus, we assume a fixed packet size per node while the adaptive packet size feature will be evaluated in future work.

IV. Slot and SF allocation algorithms

In the proposed allocation algorithms we assume that the number of nodes N as well as the amount of data they transmit is known to the gateway. Every node that requests to join the network synchronises to the gateway's clock and then goes back to sleep mode. It will wake up at a predefined time to receive the scheduling information.

A. Global allocation

In order to achieve as short data collection time as possible, the transmissions of any single node may be assigned different SFs. In this case, the time between two successive transmissions per node may not be equal, however, the 1% radio duty cycle restriction is always respected.

This approach is implemented in the Global algorithm (Algorithm 1). The algorithm takes as input the buffered data size and minimum SF for each node $(data\_size_i, minF_i)$, and the payload size per SF $payload_f$. It works iteratively, processing the list of nodes in decreasing order of their initial SF. Keeping track of the reserved slots for each SF, it assigns transmissions to the earliest unoccupied slot that also satisfies the duty cycle $(trans\_allow)$.

The algorithm generates a collection of tuples $\mathcal{S}$ consisting of the node id, the SF, and the slot number for each transmission. Because Global schedules each transmission independently the resulting schedule is not necessarily periodic and consists of a single – long – frame.

The time complexity of Global is $\mathcal{O}(6|N|\lceil\frac{data}{payload}\rceil)$ assuming that all nodes transmit the same amount of data (i.e., $data\_size$) and the payload size is the same for all SFs. The longest run occurs when all nodes can reach the gateway with SF7 so all available SFs (i.e., 6) need to be tested. In this case,

```
Algorithm 1: Global Slot and SF allocation
input : $payload_f \ \forall\ f \in [7,12], BW, guard, N,$
        $(data\_size_i, minF_i)\ \forall\ i \in N$
1  $\mathcal{S} = \emptyset$;
2  foreach $i \in N$ do
     $rem\_data_i = data\_size_i, trans\_allow_i = 0$;
3  sort $N$ in $minF$ descending order;
4  $temp\_N = N$;
5  $checked = \emptyset$;
6  while $|checked| < |N|$ do
7      if $temp\_N = \emptyset$ then $temp\_N = N$;
8      while $rem\_data_n \leq 0$ do
         $n = \texttt{shiftleft}(temp\_N, 1)$;
9      $selF = 0$;
10     $min\_t = \texttt{max int}$;
11     $min\_s = 0$;
12     foreach $f \in [minF_n, 12]$ do
13         $s = \lceil \frac{trans\_allow_n}{airtime(f,BW,payload_f)+2\cdot guard} \rceil$;
14         while $slots_f^s == 1$ do $s = s + 1$;
15         $t = 0$;
16         if $rem\_data_n \leq payload_f$ then
17             $t = (s+1) \cdot (airtime(f, BW, payload_f) +$
                 $2 \cdot guard)$;
18         else
19             $t = (s+1) \cdot (airtime(f, BW, payload_f) + 2 \cdot$
                 $guard) + 100 \cdot airtime(f, BW, payload_f)$;
20         if $t < min\_t$ then
21             $min\_t = t$;
22             $selF = f$;
23             $min\_s = s$;
24     $slots_{selF}^{min\_s} = 1$;
25     $\mathcal{S} := \mathcal{S} \cup \{n, selF, min\_s\}$;
26     $rem\_data_n = rem\_data_n - payload_{selF}$;
27     if $rem\_data_n \leq 0$ then $checked := checked \cup \{n\}$;
28     $trans\_allow_n = (min_s - 1) \cdot$
         $(airtime(selF, BW, payload_{selF}) + 2 \cdot guard) +$
         $guard + 100 \cdot airtime(selF, BW, payload_{selF})$;
29 return $\mathcal{S}$;
```

the sorting of nodes according to their minimum SF value ($minF$) can be skipped (line 3).

B. Light allocation

Unlike Global, the *Light* algorithm produces a periodic schedule consisting of repeated frames. Only the first transmission for each node is scheduled and the nodes maintain the same SF throughout the data collection. The advantage of this approach is that the schedule is shorter and computed faster.

Light takes as input the minimum SF per node, $minF_i$, and the payload size per SF, $payload_f$. It keeps track of the current (accumulated) time per SF f (i.e., $time_f$) and the number of allocated slots per SF (i.e., $slots_f$). It then allocates the transmissions of a node to the SF that leads to the earliest completion of the node's data collection, with the constraint of the duty cycle.

```
Algorithm 2: Light Slot and SF allocation
input : $payload_f \ \forall\ f \in [7,12], BW, guard, N,$
        $minF_i \ \forall\ i \in N$
1  $\mathcal{S} = \emptyset$;
2  foreach $f \in [7,12]$ do $time_f = 0, slots_f = 0$;
3  sort $N$ in $minF$ descending order;
4  while $|N| > 0$ do
5      $n = \texttt{shiftleft}(temp\_N, 1)$;
6      $selF = 0$;
7      $min\_t = \texttt{max int}$;
8      foreach $f \in [minF_i, 12]$ do
9          $t = 0$;
10         if $time_f < 100 \cdot airtime(f, BW, payload_f)$ then
11             $t = 100 \cdot airtime(f, BW, payload_f) +$
                 $airtime(f, BW, payload_f) + 2 \cdot guard$;
12         else
13             $t = time_f + airtime(f, BW, payload_f) +$
                 $2 \cdot guard$;
14         if $t < min\_t$ then
15             $min\_t = t$;
16             $selF = f$;
17     $time_{selF} = time_{selF} +$
         $airtime(selF, BW, payload_{selF}) + 2 \cdot guard$;
18     $\mathcal{S} := \mathcal{S} \cup \{n, selF, slots_{selF}\}$;
19     $slots_{selF} = slots_{selF} + 1$;
20 return $\mathcal{S}$;
```

Light's complexity is $\mathcal{O}(\mathcal{L} + |N|F_{avg})$, where $\mathcal{L}$ is the complexity of the sorting algorithm (line 3), $|N|$ is the number of nodes, and F_{avg} the average initial SF of the nodes. If, initially, all nodes can reach the gateway with SF7 then the computation of the sorting algorithm can be skipped, yielding complexity $\mathcal{O}(6|N|)$.

V. Performance Evaluation

A. Simulation Setup and Evaluation Metrics

We assume a fixed, square geographical deployment area with 1000 meters side length and a variable number of nodes that are randomly and uniformly scattered on the terrain. We compare *Light*'s and *Global*'s performance to LoRaWAN. All scenarios use unconfirmed transmissions. For LoRaWAN, the nodes use the minimum SF that allows them to reach the gateway. Their transmissions are initially delayed by an offset in $[0, airtime(SF, BW, payload_{SF}) \cdot |N|]$ to alleviate the burst of collisions. Every following transmission is performed on a periodical basis as soon as the duty cycle restriction allows it. We assume that two or more LoRaWAN transmissions collide when they overlap in time, SF, BW, and received power [6]. Non-orthogonality of SFs is taken into account. No external interference exists. To guarantee a

TABLE I
SIMULATION PARAMETERS

Parameter	Value
Nodes	10 – 1000
Terrain side (TS)	1000 m
Gateway coordinates	[TS/2, TS/2, 10] m
Bandwidth (BW) - Coding Rate	500 KHz - 4/5
Packet size	100 Bytes (unless specified)
Data per node	1024 Bytes
Path Loss model	$\overline{L_{pl}}(d_0) = 95$dB
(see [6])	$d_0 = 40$m, $\gamma = 2.08$, $\sigma = 3.57$
Capture effect isolation thresholds	same as [5]
Guard time	40 ms
Receiver Sensitivities	[-116, -119, -122, -125, -128, -129]
(per SF for BW500)	dBm
Tx power & consumption	14 dBm, 25 mA

fair comparison between the three approaches, we consider a packet size equal to 100 bytes even though it is not supported by all the SFs of LoRaWAN. For the scheduling approaches, our experimental results showed that a guard time of 40ms is enough to avoid collisions for more than 10 minutes with a single synchronisation packet.

We measure the data collection time, packet delivery ratio (PDR), and the energy consumption. We normalise the time and energy to the packet delivery ratio so that we can show the values to be expected if 100% PDR is achieved. We note that the normalised values are actually best case because in reality the collection time and energy consumption vary supra-linearly with the PDR. We generate 50 instances per scenario and the average results are presented along with the 95% confidence intervals. Table I summarises the value used for each parameter. All algorithms are implemented in the Perl programming language[1].

B. Fixed Data Size Study

In this study, all nodes have the same amount of data (1KB) to transmit, which represents a periodic application. The results are depicted in Fig. 2, where the zoom-in window presents the results from 10 to 90 nodes. The impact of the synchronisation is obvious in the PDR, where *Global* and *Light* achieve almost 100% PDR independently of network size. Without synchronisation, the LoRaWAN PDR drops quickly as more nodes are added (see Fig. 2c). This is due to the scalability issue of the ALOHA-based MAC, where nodes experience severe collisions. The poor PDR of LoRaWAN contributes also to the other two metrics and, therefore, LoRaWAN experiences the highest normalised collection time and energy consumption compared to the other two approaches.

From Figs. 2a and 2b, we can observe that *Global* achieves a slightly lower collection time than *Light*, but this comes at the expense of higher energy consumption. The reason behind this is *Global*'s ability to schedule individual transmissions on any of the *allowed* SF, which allows more concurrent transmissions and, thus, lower collection time. However, *Light* does not allow nodes to change their SFs per each transmission and thus uses low SFs most of the time, achieving the lowest energy consumption. This conclusion is supported by Fig. 3, which presents the number of slots per SF in *Global* and *Light*. Indeed, on one hand, *Light* places more transmissions in lower SF frames while SF11 and 12 are not used at all. One the other hand, *Global* can switch some transmissions to higher SFs and, thus, balance the length of the SF frames. However, this action increases the average airtime which results in higher energy consumption.

[1]The code is available at https://github.com/deltazita/offline-lora

C. Variable Data Size Study

In this study, the nodes start the data collection with variable data sizes, which represents an event-based application. The data size of each node is randomly allocated from $[1000 - X/2, 1000 + X/2]$ bytes, where $X = [100..500]$. The data collection time of this study is depicted in Fig. 4. The energy consumption and the PDR are not presented because they are almost the same as the respective results in the fixed data size study. Fig. 4 shows the advantage of *Global* over *Light* by achieving significantly less data collection time. However, the trend here is clearer than in the fixed data sizes study.

Because there is variation in the amount of data available per node, some nodes will finish transmitting sooner than others. The impact on the two algorithms is different. First the *Light* schedule is done based on the first transmission and regardless of the amount of data per node. As a result, nodes that complete sooner leave empty, unused slots. This becomes inefficient when the frame is longer than the duty cycle, in which case those slots could be used instead by other nodes' transmissions. The *Global* schedule does not have this problem, because all transmissions are scheduled. If a node finishes early the slots will remain free and will be allocated to subsequent transmissions. The *Global* algorithm packs the schedule tightly with few unused slots, achieving a schedule length (over all SF rows) that is close to the total amount of transmissions in the network. Therefore, as the total data in the network is on average 1000B in all the iterations, *Global* achieves a roughly constant length schedule. In contrast, the data collection in *Light* will only finish when the node with the most available data finishes.

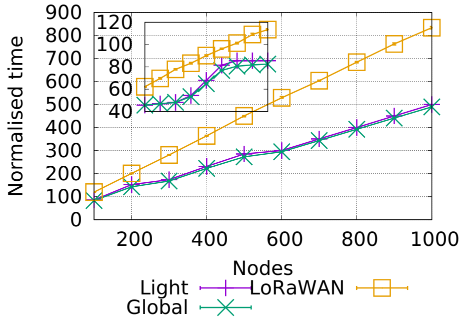
D. Variable Packet Sizes Study

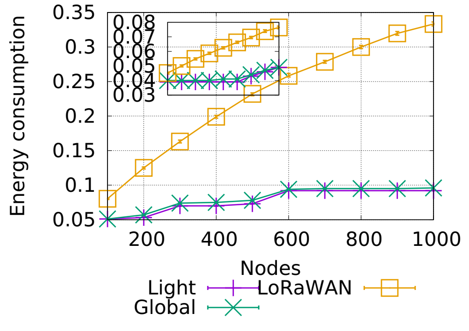
Here, we assess the effect of the packet size on the normalised data collection time, the normalised energy consumption, and the PDR. The results are obtained from a network of 500 nodes and are presented in Fig. 5. They show a considerable improvement for the first two metrics when the packet size increases, due to a higher data to header ratio. Both *Global* and *Light* present the same trend as the packet size increases. LoRaWAN's PDR presents an almost constant trend due to a similar transmission time – number of transmissions ratio in the system.

VI. DISCUSSION

A. Transmission of the schedule

In the following we discuss schemes for transmitting the schedule from the gateway to the nodes, which constitute a significant time overhead due to the duty cycle regulations on the downlink channel.


(a) Normalised Time


(b) Normalised Energy

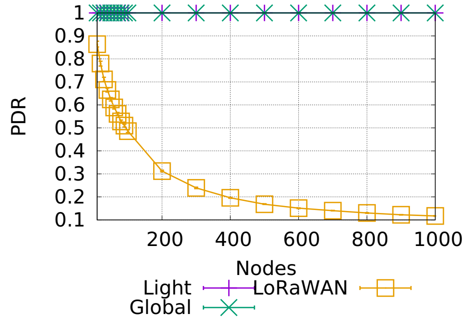

(c) Packet Delivery Ratio

Fig. 2. Normalised data collection time, normalised energy consumption, and Packet Delivery Ratio for a scenario with variable number of nodes.

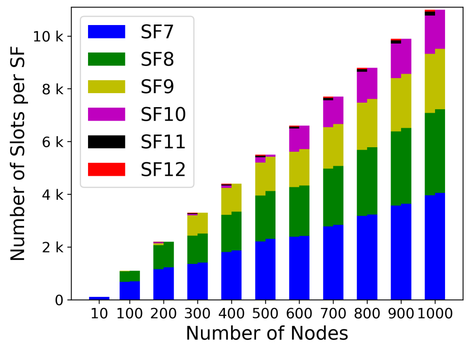


Fig. 3. Number of reserved slots per SF for Global (first bar) and Light (second bar).

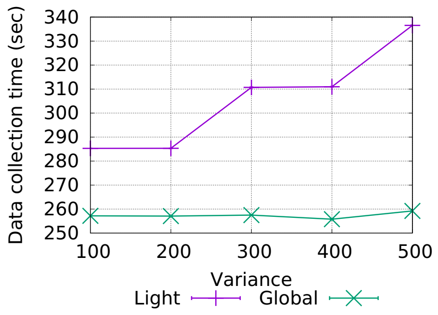


Fig. 4. Data collection time for different transmitted data sizes per node.

In the schedule generated by the Light algorithm each node must know on which SF row and at which slot offset it has to transmit. We transmit this information (node id, SF row, offset) to nodes, grouped by their minimum SF. By bundling the information we reduce the MAC packet overhead. Furthermore, we can avoid sending the slot offset by sending the node id's in the order that they are scheduled in. Note that to minimise the schedule length some nodes are bumped up into higher SF rows. The list transmitted to nodes with minimum SF f is therefore split in blocks, corresponding to the nodes allocated to SF frame f, $f+1$, etc. Each block requires an additional three fields: the index in the transmitted list (2B), the offset in the allocated SF row (2B), and the length of that row (2B). To account for resulting packets longer than the limit (255B), a fragment ID (1B) is included. As a result, for m_f nodes with minimum SF f the gateway will send $(12-f)\cdot 6+2\cdot m_f$ (plus 1B fragment ID per packet). The overhead in time is shown in table II.

In the Global algorithm the schedule for a node must describe each individual transmission, so the schedule length (per node) depends on the amount of data available. Communicating such a schedule is not scalable. On the other hand, the algorithm input only consists of the list of nodes, their data size, and minimum SF. We propose to send this information to the nodes and let them instead run the algorithm. As indicated in Section IV-A the algorithm's complexity is linear to quadratic (depending on the amount of data), which is not restrictive even for performance constrained devices [2]. To reduce the complexity at the nodes, the gateway will communicate the list of nodes already sorted. We further point out that the nodes must only run the algorithm for all the nodes in the sorted list up to and including themselves. Then, for each set of nodes on the same minimum SF f the gateway will send: 1B fragment id; for each higher SF, $f+i$, the number of nodes with the particular minimum SF and data size for those nodes.

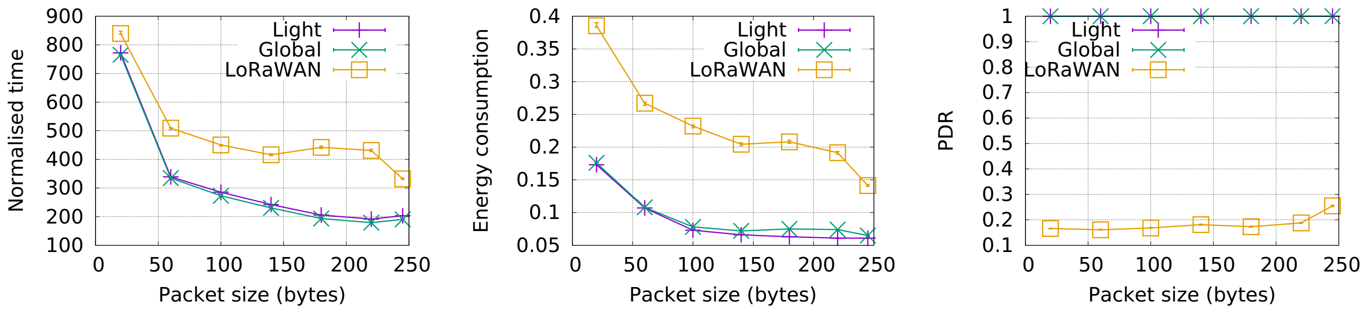
The results shown in Table II demonstrate that the transmission of the schedule constitutes a significant overhead to the data collection, which diminishes the advantages of the collision free communication. With this overhead accounted for the *Light* algorithm is clearly the better solution in both time and energy consumption.

The schedule transmission overhead is part of a fundamental problem in LoRa networks. The system is inherently centralised, precluding decentralised algorithms, however, the central point is a communication bottleneck. This is a paradox, and using hybrid centralised-decentralised algorithms, as proposed for the global algorithm, may be an interesting solution.

B. *Global vs Light scheduling*

Global has the capability of manipulating both dimensions of the 2D frame array at any time during the allocation process. This means that successive node transmissions can be carried out on different SFs and fill the empty slots caused when nodes finish transmitting sooner than others. This has

[2]It could be argued that the actual bottleneck is memory, which varies linearly with the network size.



(a) Normalised Time (b) Normalised Energy (c) Packet Delivery Ratio

Fig. 5. Normalised data collection time, normalised energy consumption, and Packet Delivery Ratio for a scenario with variable packet size and 500 nodes.

TABLE II
GATEWAY DOWNLINK TIME

Nodes	Transmission time (s)		Nodes	Transmission time (s)	
	Global	Light		Global	Light
10	0.03	0.02	600	29.25	8.48
100	1.53	0.66	700	33.29	9.75
200	4.46	1.08	800	36.96	10.87
300	10.82	2.88	900	41.06	17.52
400	13.23	3.85	1000	50.20	18.76
500	18.76	7.53			

an obvious effect on the data collection time but comes with the expense of consumed energy since transmissions are, on average, reallocated to higher SFs. However, as discussed in the previous subsection, the main drawback of *Global* is the size of the schedule that depends both on the number of nodes and the data size. *Light* does not encounter *Global*'s scalability issue since the schedule length is independent of the data size. This comes with the expense of a higher collection time, however, as the simulated results showed, the difference with *Global* is slight for periodic applications.

In conclusion, the algorithms' practicality depends on the examined scenario. Assuming that both approaches are executed at the gateway, *Light* seems to be the best option for large-size periodic applications, while *Global*'s usage is limited to event-based applications where the time needed to send the schedule to the nodes can be covered by the reduced data collection time.

VII. Conclusion & Future work

In this paper we presented two offline scheduling algorithms for LoRa-based data transmissions. Both approaches allocate time slots and assign SFs to nodes aiming at minimising the total data collection time. The simulation results showed, on one hand, that *Global* presents a slight to large improvement over *Light* in terms of collection time for periodic and event-based applications, respectively. However, this comes with the expense of the higher energy cost and the longer schedule. On the other hand, *Light* tends to keep the nodes on lower SFs achieving a lower energy cost. Finally, the comparison with LoRaWAN showed a large performance improvement over the native protocol for all considered performance metrics.

In the future, we will assess *Global* and *Light* considering a confirmable application as well. We will also investigate whether a different packet size per SF can further improve the data collection time without compromising the energy cost and the network reliability.

Acknowledgements

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) 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] J. Haxhibeqiri, E. De Poorter, I. Moerman, and J. Hoebeke, "A survey of lorawan for iot: From technology to application," *Sensors*, vol. 18, no. 11, 2018.

[2] 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.

[3] S. Kim, R. Fonseca, P. Dutta, A. Tavakoli, D. Culler, P. Levis, S. Shenker, and I. Stoica, "Flush: a reliable bulk transport protocol for multihop wireless networks," in *SenSys*. ACM, 2007, pp. 351–365.

[4] ETSI ERM TG28, "Electromagnetic compatibility and radio spectrum matters (erm); short range devices (srd); radio equipment to be used in the 25 mhz to 1000 mhz frequency range with power levels ranging up to 500 mw," *European harmonized standard EN*, vol. 300, no. 220, p. v2, 2012.

[5] D. Magrin, M. Centenaro, and L. Vangelista, "Performance evaluation of lora networks in a smart city scenario," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–7.

[6] 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.

[7] K. Mikhaylov, J. Petäjäjärvi, and J. Janhunen, "On lorawan scalability: Empirical evaluation of susceptibility to inter-network interference," in *EuCNC*, June 2017, pp. 1–6.

[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] M. Rizzi, P. Ferrari, A. Flammini, E. Sisinni, and M. Gidlund, "Using lora for industrial wireless networks," in *IEEE 13th International Workshop on Factory Communication Systems (WFCS)*, May 2017, pp. 1–4.

[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.