LoRaWAN Scheduling: From Concept to Implementation

Celia Garrido-Hidalgo, Jetmir Haxhibeqiri, Bart Moons, Jeroen Hoebeke, Teresa Olivares, F. Javier Ramirez, and Antonio Fernández-Caballero

***Abstract*—While the Internet of Things continues to grow, the LoRaWAN standard is generating special interest due to its open-source nature, ultra-low power consumption and long-range connectivity. Recent works have explored the challenges of implementing LoRaWAN, with scalability being considered one of the major bottlenecks imposed by its Aloha-based MAC layer. Despite much on-going research on LoRaWAN scheduling aimed at alleviating this concern, experimental approaches are rarely found in the literature. In this work, we describe the steps taken and the technical issues overcome to move from a low-overhead synchronization and scheduling concept to its real-world implementation on top of LoRaWAN Class A. Accordingly, an end-to-end architecture was designed and deployed on top of STM32L0 MCUs, which communicate with a central entity responsible for providing synchronization metrics and allocating transmission slots on demand. The clock drift of devices was measured in a temperature-controlled chamber, which served as a basis to define slot lengths in the network. As a result, an operational end-to-end system was implemented and evaluated for different setup scenarios, with 10-millisecond accuracy being achieved. Our experimental results show significant improvements in packet delivery ratios with respect to Aloha-based setups, especially under high network loads (up to 29% for SF12), thereby demonstrating the feasibility of the presented approach.**

***Index Terms*—LoRa, LoRaWAN, scheduling, synchronization, clock skew, clock drift.**

I. Introduction

As the Internet of Things (IoT) continues to expand, forecasts point to 22 billion connected devices by 2025 (excluding daily-life gadgets), of which nearly 2 billion will be fully operated by Low Power Wide Area Networks (LPWANs) [1]. In the LPWAN domain, LoRaWAN [2], which leverages ultra-low power consumption and long-range communication on top of Semtech's LoRa PHY layer [3], has attracted special interest. Similarly to how Wi-Fi technology is considered the major privately-operated competitor for cellular technologies, LoRaWAN has become a feasible private alternative to other LPWANs. While being highly suitable for battery-powered devices, it also features low deployment and operational costs by using the unlicensed frequency bands.

C. Garrido-Hidalgo, T. Olivares and A. Fernández-Caballero are with Instituto de Investigación en Informática de Albacete and Departamento de Sistemas Informáticos, Universidad de Castilla-La Mancha, 02071 Albacete, Spain (e-mail: celia.garrido@uclm.es).

J. Haxhibeqiri, B. Moons and J. Hoebeke are with IDLab, Department of Information Technology at Ghent University - imec, 9052 Ghent, Belgium.

F.J. Ramirez is with Escuela Técnica Superior de Ingenieros Industriales, Departamento de Administración de Empresas, Universidad de Castilla-La Mancha, 02071 Albacete, Spain.

Being oriented towards infrequent machine-type communication, LoRaWAN relies on Aloha, which guarantees cost-effective communications by avoiding Medium Access Control (MAC) overhead [4]. While suitable under low traffic loads, reliability is significantly compromised in large-scale deployments leading to a network collapse [5], [6]. To tackle this, several scheduling schemes exist but most have only been validated in simulators, thus not considering the issues related to implementing an operational end-to-end system.

To fill this gap, this work extends a recent contribution [7] where a low-overhead synchronization and scheduling technique was presented. Slots were assigned on demand by the Network Synchronization and Scheduling Entity (NSSE) using space-efficient probabilistic data structures. Synchronization energy overhead resulted to be much lower than the required to retransmit collided packets in unsynchronized setups.

The current work addresses a detailed implementation and experimental validation, highlighting the effort required to move from simulation to a real end-to-end system. To this end, the system architecture is first proposed and deployed, with the hurdles overcome being discussed as design choices. Furthermore, a methodology was designed to measure the clock drift of devices in different environmental conditions, which served as a basis to define guard times mitigating the effect of device desynchronization over time. The resulting system was evaluated for high network loads in terms of synchronization accuracy and communication reliability.

To the best of our knowledge, this is the first end-to-end solution providing experimental clock drift measurements and validation of a LoRa-based synchronization and scheduling scheme. As major contributions, first, a measurement methodology is designed and validated under temperature-controlled conditions. Second, the design choices and hurdles overcome are described in-depth, including: clock source selection, architecture constraints, error compensation, traffic priorities, and asymmetric network delays, among others. Finally, the end-to-end system is validated on top of LoRaWAN hardware, with a mean synchronization accuracy of 10 milliseconds, and delivery ratios close to 100% under high network loads, that is, 29% higher than those of unsynchronized setups.

The remainder of this paper is structured as follows. Section II reviews LoRaWAN and related works. In Section III, the architecture and algorithm's fundamentals are detailed. Section IV describes the implementation process, challenges and lessons learned. Section V presents the experimental evaluation of the system and discussion. Lastly, Section VI provides the conclusions and explores the future research.

II. Background

A. LoRa and LoRaWAN

LoRaWAN [2] defines the MAC-layer for LoRa's PHY-layer by Semtech [8]. LoRa is based on a proprietary spread spectrum modulation derived from Chirp-Spread Spectrum (CSS), where data rates (DRs) depend on the spreading factor (SF), bandwidth (BW) and coding rate (CR). The SF can be set to six values (SF7 to SF12), with 2^{SF} being the number of chirps per symbol. Each chirp consists of a linear frequency sweep covering a typical BW of 125 kHz (*up-chirps* for increasing frequencies and *down-chirps* for decreasing ones). LoRa signals using different SFs are quasi-orthogonal and, hence, not expected to interfere with each other when transmitted over the same channel and BW. The lower the SF, the higher is the DR, thereby shortening packet air times at the expense of a more limited range. Conversely, higher SFs allow the receiver to discern signals with lower Signal-to-Noise Ratios (SNR). LoRa modulation performs forward error correction, which increases redundancy at the expense of longer transmission air times. The CR is computed as the data-word length (k, equal to 4) divided by the code-word length (n, in the range [5, 8]), which is set to 4/5 by default. Higher n values will therefore improve robustness against interference.

The LoRa packet comprises the preamble, an optional header and the frame payload. The time on air of a LoRa frame is based on the length of these fields and symbol duration, according to formulae provided in [3]. The preamble is used to synchronize the receiver with the transmitter, which consists of an optional programmable part followed by four fixed symbols encoding the synch word. As long as the receiver and transmitter have similar Signal Strength Indicators (RSSIs), the reception of the last six symbols of the preamble typically suffices for the receiver to synchronize [9].

LoRaWAN operates in the unlicensed frequency bands of 433, 868 and 915 MHz. In Europe, there are three mandatory 125 kHz channels (868.1, 868.3 and 868.5 MHz) [10]. It uses a star-of-stars topology, where gateways are responsible for forwarding uplink frames from end devices to a network server via a back-haul, such as Ethernet or 4G. Downlink communication is also possible but not encouraged to sustain scalability, given the half-duplex nature of gateways and the duty-cycle limitations [11]. The duty cycle is the maximum time percentage a device can occupy a channel, and is regulated by the EN300.220 standard [12] in the 868-MHz band.

The standard defines three classes of end devices according to the application being pursued. Class A, supported by any LoRaWAN device, achieves the lowest power consumption by opening two receive windows only upon uplink completion (after 1 and 2 seconds, respectively). This behavior allows the end device to save energy by remaining in low-power modes for long periods of time but, consequently, downlink traffic should be accordingly scheduled at the network-server side.

The LoRaWAN channel access strategy relies on pure Aloha, with devices transmitting asynchronously without performing any carrier sensing [4]. This has some disadvantages in terms of reliability, especially when high network loads exist, as collision rates greatly increase [13].

B. LoRaWAN scheduling

While there is much on-going research on LoRaWAN scheduling aimed at alleviating related scalability concerns, only a few experimental approaches exist in the literature.

Reynders *et al.* [14] presented a two-step scheduling scheme, validated through the NS-3 simulator, where end devices select their desired channel, SF and transmission power based on information provided by gateways on demand. Similarly, Abdelfadeel *et al.* [15] used the LoRaFREE simulation tool to demonstrate the benefits of allocating simultaneously transmissions over different SFs and sequentially single-SF ones, for which a two-step synchronization mechanism was implemented and evaluated. While these approaches fully rely on LoRaWAN Class A, Rajab *et al.* [6] provided a Matlab simulation model where end devices synchronize over Class C and, immediately, switch to Class A to save energy. In this case, SFs are assigned by gateways based on the distance to end nodes. Finally, a Time-Division Multiple Access (TDMA) scheme built on top of Class B is presented in [5]. In this work, the so-called *Class-S* is evaluated in terms of energy efficiency and throughput compared to legacy Class A using the *LoRaWAN-Sim* ad-hoc simulation environment.

The scheduling concept presented in [7], whose real-world implementation is pursued in this work, fully adheres to the LoRaWAN standard. It is built on top of Class A specification while most LoRa-based scheduling works require modifications of the standard itself or rely on different classes. For instance, the authors in [14] include beacons for synchronization purposes, which breaks the concept of using Class A devices. In [15] and [6], the use of an initial broadcast phase at the gateway-side might overuse the duty-cycle limitations imposed to gateways. Finally, the so-called *Class S* presented in [5] fully operates on top of Class B, which is not mandatory to be implemented by LoRaWAN-compliant end devices.

To date, some studies addressing the real-world evaluation of LoRaWAN applications exist in the literature. By covering different areas such as agriculture [16], urban monitoring [17], supply chain management [18], or earthquake detection [19], these typically evaluate the performance of legacy LoRaWAN with no meaningful improvements being provided on top of its MAC layer. Only two experimental approaches addressing LoRa-based scheduling have been presented in the literature to date. In the first, Zorbas *et al.* [20] proposed dividing time frames in slots, which were pre-assigned to end devices based on their extended unique identifier (EUI). The proposal was validated in a LoPy4 testbed. Their mechanism consisted of using a dedicated slot per frame to handle synchronization and acknowledgments. However, the synchronization signaling overhead led to a 50% increase in energy consumption, which should be accordingly addressed. The scheduling mechanism pursued in this work, conversely, requires no additional radio interfaces other than LoRaWAN. The second experimental approach [21] consisted of a LoRa on-demand TDMA scheme, which was evaluated experimentally in a testbed of 11 MSP430-based nodes. Although scheduling was targeted for LoRa communication, the nodes were equipped with a short-range receiver based on PIC12LF1552 Microcontroller Units

(MCUs) to perform synchronization with 0.1-ms accuracy and initiate the slot-allocation mechanism. This was done through a cluster head, also consisting of a dual-radio interface to enable short-range communication with end devices and long-range with gateways. This solution, however, considerably increases the complexity of the hardware required. The target scheduling approach was evaluated in this sense, where the authors demonstrated a low-overhead feature with respect to other approaches where gateways need to perform an initial broadcast phase to schedule end nodes. For the aforementioned reasons, moving from the scheduling concept presented by Haxhibeqiri *et al.* [7] to its real-world implementation is one of the main motivations in this work.

It is important to note that, in order to successfully address time-division scheduling approaches in practice, both synchronization accuracy and clock-drift measurement are key. Even if two clocks were accurately synchronized in time, differently-drifting clocks will provoke their desynchronization over time. Clock drift, however, received scant attention in most studies despite being an important design aspect to this end. In fact, none of them provides any measurements in this regard, typically assuming ± 15 to ± 30 ppm for all devices in the network, which might be unrealistic in real large-scale deployments. Furthermore, network latencies and hardware-related limitations are usually neglected when using simulation environments, whose influence can hinder the performance of scheduling proposals that need to rely on synchronization. This work addresses the practical synchronization and scheduling of Class A end devices, providing and validating a clock drift measurement methodology to alleviate desynchronization concerns arising in real-world network deployments.

C. Clock synchronization over LoRaWAN

First, we present some key notions on clock synchronization. The behavior of a clock over time can be characterized based on its *offset*, *skew*, and *drift*, used hereafter following the terminology provided in [22] and defined as follows:

- Clock offset: time difference between two clocks.
- Clock skew: difference between a given clock's frequency and its nominal value, f_0. It is the first derivative of a clock value with respect to time and provides a measure of how much a clock is incremented per time unit. *Clock skew* can also be referred to as *clock rate*.
- Clock drift: difference in rates of a given clock and a reference clock (second derivative), usually due to environmental factors such as temperature, humidity, pressure, and orientation. Of these, quartz crystals are known to present the highest sensitivity to drastic temperature changes [23], which presents a parabolic frequency dependence over time given by:

$$\Delta f / f_0 = \beta \cdot (T - T_0)^2 \tag{1}$$

  where β is a temperature coefficient ($ppm/^\circ C^2$), and T_0 the turnover temperature in the range $25 \pm 5^\circ C$.

Clock synchronization, despite being a challenging task due to low-cost embedded oscillators and environmental uncertainties, is known to enhance the reliability of large-scale LoRa networks. In [24], a synchronization mechanism used to develop a slotted-Aloha LoRaWAN variant was presented, with transmission success rates improving from 7% to 33%.

Different LoRa-based synchronization approaches are found in the literature. Polonelli *et al.* [25] achieved a mean accuracy of 10 milliseconds using STM32L4 MCUs by computing a received timestamp and the processing and air-times elapsed. A similar method was conducted by Tessaro *et al.* [26] on top of STM32L073 MCUs. Despite reaching ± 4.5-millisecond synchronization accuracies, clock-register resolution was limited to 1024 ticks per second. Using the same hardware platform, the authors in [27] presented a GPS-based proposal, where the desired synchronization accuracy threshold, considering the uncertainty caused by frequency instability of oscillators, was achieved by adapting resynchronization periods. For instance, resynchronization was performed every 70 seconds if less than 13-millisecond accuracy was needed, at the expense of higher power consumption. Likewise, the authors in [28] achieved a 0.3-millisecond accuracy on top of STM32L476RG, using DCF77 and GPS timestamping. In this case, synchronization was carried out through beacon flooding in a LoRa mesh network testbed, thus not prioritizing low-power operation.

While none of the previous studies considers clock desynchronization over time, a clear trade-off between achievable accuracy and energy consumption is present in all of them. The use of power-demanding hardware such as GPS is, therefore, only justifiable for strict real-time applications.

III. LoRaWAN Scheduling Architecture

A. Network architecture

The proposed end-to-end architecture is depicted in Fig. 1. For the end-device logic implementation, we selected B-L072Z-LRWAN1 discovery kits, which include a 32-bit Arm-based STM32L072CZ MCU (192 KB Flash, 20 KB RAM, 6 KB EEPROM) and an SX1276 transceiver on a stand-alone module. These devices were programmed with the OSS7 open-source software stack, governed by a flexible framework that comprises a lightweight scheduler [29]. The stack follows a modular design and supports different communication standards, of which Semtech's LoRaWAN stack v1.0.2 was used to implement the desired end-device functionality.

Data packets are forwarded from end devices to the LoRaWAN backend, through a multichannel gateway. ChirpStack is an open-source backend that features a network server, an application server, and a gateway bridge that communicates with Semtech's UDP packet forwarder [30].

The NSSE functionality was implemented using the Click Router framework [31], which subscribes to ChirpStack's MQTT broker to receive synchronization requests and publishes replies to be forwarded to end devices. Hence, data packets received via MQTT go through processing modules (so-called *elements*) responsible for classifying packets, processing requests and scheduling transmissions from all devices joining the network based on their communication requirements. To this end, the five baseline elements at the back-end side are: (i) MQTT subscriber, (ii) packet classifier, (iii) data-packet manager, (iv) NSSE, and (v) MQTT publisher. Further implementation details are provided in Section IV-A.

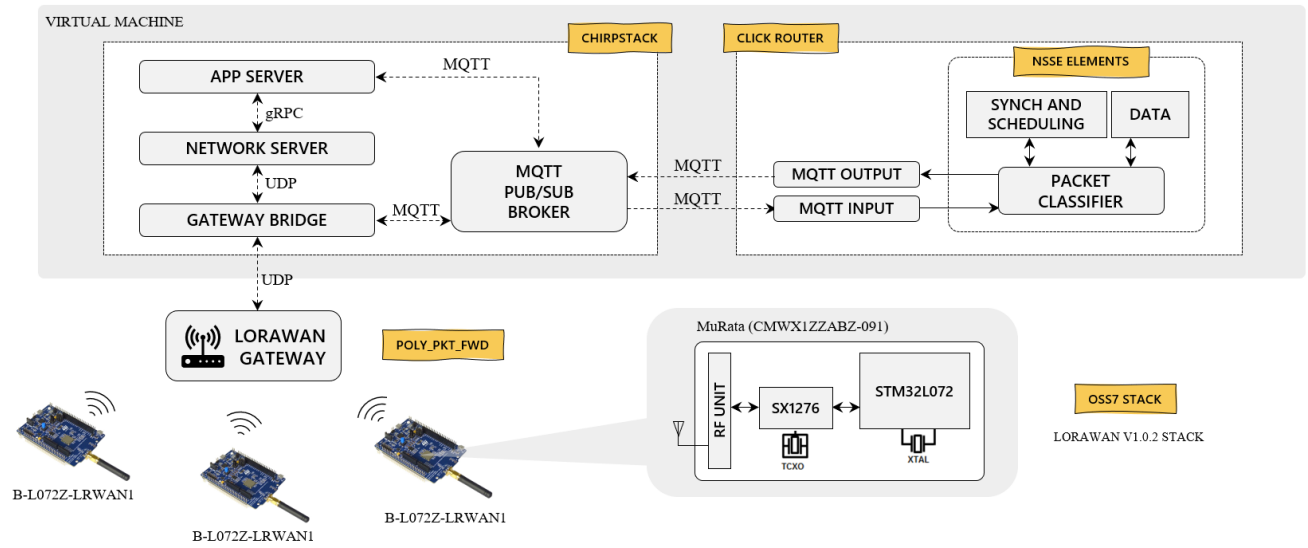


Fig. 1. End-to-end LoRaWAN scheduling architecture.

B. Synchronization and scheduling algorithm

The fine-grained synchronization and scheduling scheme proposed is built on top of LoRaWAN MAC layer, intended for Class A devices that trigger communication. Although the algorithm was presented in [7], some design aspects of the initial algorithm were changed to deal with hardware limitations encountered during implementation (further addressed in Section IV). Fig. 2 shows a finite state machine describing the set of stages that LoRaWAN devices go through once they initiate the on-demand synchronization mechanism.

Devices joining the network send a synchronization request to the NSSE including their: (i) clock skew, (ii) request identifier, (iii) uplink periodicity needs, and (iv) resynchronization period. To filter MAC packets at the network server, port 224 is reserved for synchronization traffic (both uplink and downlink) while a generic port is used for periodic communication. These ports are configurable in the setup. Likewise, synchronization can be performed in-band or out-of-band. In the former, synchronization traffic shares the three LoRaWAN mandatory standard channels with uplink data while, in the latter, the 869.525 MHz channel (only used for downlink traffic at receive window RX2) is reserved for synchronization (typically known as high-power (500 mW), high-duty cycle (10 %) channel [12]). If the network overloads, the NSSE will not allocate more devices to prevent packet losses. In this case, devices attempting to join in band using a 0.1-% duty cycle will have a low impact on the network performance while out-band synchronization will not have any impact.

The NSSE allocates time slots able to accommodate the highest time-on-air in the network, using two mechanisms:

1) Fixed slot allocation: each device has a set of pre-assigned slots that are hashed by the NSSE based on their EUI. The payload includes the first transmission slot (after which the device will follow its own periodicity) and a set of forbidden slots (where other devices in the network are expected to perform resynchronization).
2) Bloom filter allocation: it consists of using probabilistic data structures where available slots are encoded and decoded applying a double-hashing technique [7]. This method, despite being a space-efficient alternative, has a false-positive probability given by:

$$p \approx \left(1 + e^{-\frac{k \cdot n}{m}}\right)^k \tag{2}$$

where p is the false-positive probability, k the number of hash functions, n the number of entries, and m the filter size (bits).

Once the request is sent, the end device maps its local timestamp with its identifier and schedules the next synchronization retry (see Fig. 2). It then remains idle until a downlink is received, which goes through a double-filtering stage. Otherwise, the scheduled request will be triggered.

During the first filtering stage, the end device parses the frame port and the packet identifier to check whether it corresponds to the last request sent. Otherwise, it will automatically discard the received downlink at hand and remain idle until the next-scheduled retry. Since devices operate Class A, delayed downlinks unable to reach any of the receive windows will remain buffered at the network server following a FIFO queue (addressed in Section IV). Thus, received packet identifiers can be either equal to, or lower than the last-sent request.

Once the first filtering is performed to ensure identifiers match, the device parses the remaining fields: (i) slot index, (ii) offset at current slot index, (iii) resynchronization slot, and (iv) available transmission slots. Assuming that network delays are symmetric and negligible, the device can synchronize its clock with the global timestamp provided by the NSSE. To do this, the device computes the slot index (S_{i+n}) and slot offset (δ_{ed}) at current time by considering the local time elapsed since the identified request was sent, timestamp provided by the NSSE, and uplink time on air consumed. An explanatory time diagram showing the synchronization process between the end node and the NSSE can be accessed in Fig. 2 from [7]. Calculations are made according to Eqs. (3) and (4):

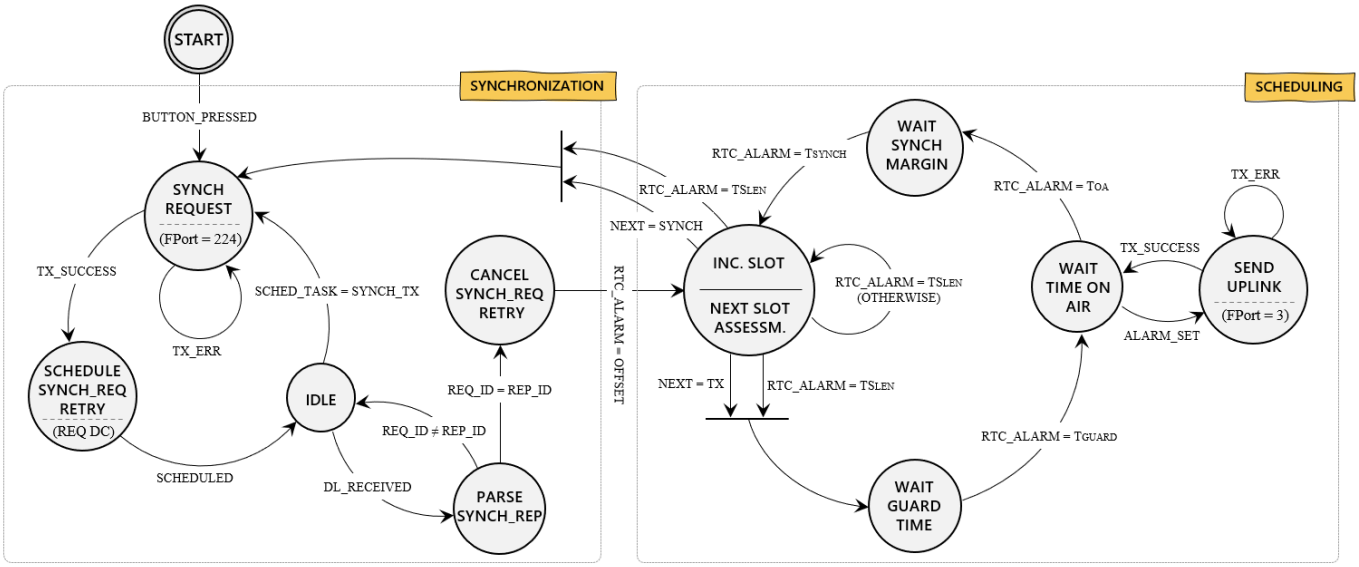


Fig. 2. Synchronization and scheduling finite state machine.

$$S_{i+n} = S_i + (\Delta t - t_{OA} + \delta_{nsse}) \div TS_{len} \tag{3}$$

$$\delta_{ed} = \Delta t - t_{OA} + \delta_{nsse} - S_{i+n} \cdot TS_{len} \tag{4}$$

where S_i is the slot index at the NSSE, Δt is the time elapsed since the request, t_{OA} the uplink time on air, TS_{len} the slot length, and δ_{nsse} the slot offset at the NSSE.

Once the device is synchronized with the NSSE, slots are continuously incremented based on Real-Time Clock (RTC) alarms. These are conceived as down-counters able to wake up the MCU from to execute certain tasks. At every time slot, the next slot is assessed to trigger uplinks in the assigned slots, prioritizing resynchronization needs. If a transmission slot is identified, the device will split the slot length, TS_{len}, into three lengths as shown in (5): guard time (t_{guard}) based on its clock skew preventing overlaps between adjacent slots (for the specified resynchronization period), time on air (t_{OA}) required for the longest transmission in the network, and synchronization margin (t_{synch}) to mitigate the clock offset.

$$TS_{len} = t_{guard} + t_{OA} + t_{synch} \tag{5}$$

The guard time length is based on the clock skew of all the devices in the network, specifically, on the fastest and slowest clocks. Assuming no guard time at the beginning of synchronized slot boundaries, the worst-case scenario (highest overlap of transmissions) depends on whether these clocks drift in the same time direction or not. In the first case, the highest-possible overlap will take place at the end of the synchronization period, with devices simultaneously synchronizing with the NSSE. This overlap, will ideally be equal to the difference between the fastest and the slowest clocks. Conversely, when all the clocks in the network drift in the same direction, the highest overlap is also dependent on the synchronization offset between them. In this case, the worst scenario occurs when the device with the lowest drift in absolute value synchronizes prior to the device with the highest one. The resulting maximum relative drift is given by (6), maximized when t_δ tends to zero.

$$\Delta C_{max} = \max\left[C_i(T_{synch})\right] - \min\left[C_i(t_\delta)\right] \tag{6}$$

where ΔC_{max} is the maximum relative drift, $\max[C_i]$ the maximum clock skew, T_{synch} the synchronization period, $\min[C_i]$ the minimum clock skew, and t_δ the time offset between synchronization events. ΔC_{max} is part of the implementation-related design choices that are introduced in this work as an improvement with respect to the algorithm in [7]. Not considering ΔC_{max} in a real-world deployment would imply a certain offset between devices synchronizing at different times and drifting differently in time.

IV. Implementation and Lessons Learned

A. Hardware deployment

In order to validate the proposed algorithm on top of real LoRaWAN hardware, a baseline setup with four B-L072Z-LRWAN1 discovery kits was deployed (hereafter referred to as devices A-D). These development boards are based on the open CMWX1ZZABZ-091 Murata IC, which features ultra-low-power STM32L072CZ MCU (Arm Cortex M0+ core), a Semtech's SX1276 transceiver and two external clock sources, namely XTAL and TXCO (Temperature Compensated Crystal Oscillator). It is important to note that the four Murata modules used for the deployment belong to different series, which means that manufacturing dates and conditions may differ and, hence, influence experimental results. Specifically, each pair of devices had a different inspection number: SS8516005 for devices A and C (manufactured in 2018, 005 serial no) and SS7N25009 for devices B and D (manufactured in 2017, 009 serial no). Please consult to Murata's packaging reference specification for further information [32].

Once devices are synchronized with the NSSE, time is incremented on a slot basis. To do this, two alternatives were considered: (i) using the RTC to trigger alarms on slot completion, and (ii) using the stack scheduler (system clock) to register a task at every slot boundary. While the maximum operational framework resolution is 1024 ticks per second, the RTC register consists of a 15-bit down counter (32768 ticks per second) that enables higher accuracy. Furthermore, the reliability of the RTC is expected to be higher since the stack scheduler is shared by multiple resources in the system. However, RTC-based slot transitions can considerably increase the load of the system, generating time delays that depend on the size of the interruption routine executed.

As an alternative, we opted for a hybrid solution: time boundaries were implemented using consecutively-triggered RTC alarms and, once the interrupt request was triggered, a callback was used to register a maximum-priority task in the framework scheduler to avoid blocking the algorithm execution. Moreover, non-blocking resynchronization requests in compliance with duty cycle were implemented using the stack scheduler.

The RTC clock source selection was one of the key design choices, which can be sourced by LSI (low-speed internal), LSE (low-speed external) or HSE (high-speed external) oscillators. While the LSI has limited accuracy for synchronization purposes (-10% to 4%), the HSE oscillation frequency ranges from 1 MHz to 25 MHz, which greatly increases power consumption. As a result, the LSE crystal was selected to source the RTC as a trade-off between power consumption (ultra-low) and accuracy ($\pm$0.002% or $\pm$20 ppm). The selected clock source can trigger wake-up events from both stop and standby modes, achieving the lowest-possible MCU power consumption. However, this oscillator is designed and assembled by Murata, who reserves the bill of materials as part of their Intellectual Property. The fact that no characteristic curves are provided makes clock drift measurement a key challenge in the LoRaWAN scheduling scheme implementation.

The LSE oscillator has a nominal frequency of 32.768 kHz, where programmable 7-bit asynchronous and 15-bit synchronous prescalers were used to reach a trade-off between speed and current consumption [33]. With the latter being responsible for determining the sub-second RTC register resolution, different configurations were tested with occasional time inaccuracies being found for high prescaler values. As a result, the maximum RTC resolution achieved was 8192 ticks per second (13-bit prescaler), which enables a granularity of 0.122 ms. It is important to note that LoRaWAN timing events in the LoRaWAN MAC implementation used [34] are based on RTC alarms (Alarm A). Thus, a new RTC alarm (Alarm B) needed to be defined through the Hardware Abstraction Layer (HAL) drivers to separately manage scheduling events, thus not interfering with the LoRaWAN stack's functionality.

While timing events at end devices were based on RTC ticks to avoid conversion errors caused by fixed-point arithmetic, NSSE timestamping was done in milliseconds. Hence, a time-compensation mechanism was implemented in order to mitigate accumulated errors causing desynchronization over time. To do this, devices are provided with their required compensation period according to the SF used, after which an extra RTC tick needs to be subtracted or added (depending on whether they are ahead or behind of schedule, respectively).

Finally, the LoRaWAN gateway used is based on the multichannel iC880-A concentrator (SX1301 transceiver) and serves as host where the packet forwarder runs to transmit RF packets to the network server and vice versa. At the gateway-side, minor modifications were made to the LoRa concentrator HAL to enable listening to a newly-defined high-power high-duty cycle channel centered at 869.525 MHz.

B. Backend implementation

ChirpStack's network server v3.8.0 was installed in the same virtual machine where Click Router framework was deployed to minimize network delays affecting the evaluation. The NSSE resides at Click Router and keeps track of the reference timestamping used to synchronize all devices in the network. The architecture of the developed central entity consists of the following elements, which are sequentially interconnected:

- MQTT subscriber: subscribes to the application server and pushes base64-decoded data to the packet classifier. Pushed data consists of rx metadata (data rate, bandwidth, channel, RSSI, SNR, etc.) and LoRaWAN PHY Payload frames including device address, frame port and frame payloads with synchronization-request parameters.
- Packet classifier: separates synchronization requests from data traffic based on frame ports, pushing payload frames to one of the following elements:
  - Data manager: processes periodic uplink packets from devices and serves as data sink in the NSSE.
  - Synchronization and scheduling manager: keeps track of slot indexes, assigns available slots on demand, reserves resynchronization slots based on time-on-air requirements, and pushes synchronization reply fields.
- MQTT publisher: publishes base64-encoded synchronization replies to the LoRaWAN application server.

C. Challenges overcome

Once the baseline end-to-end deployment was carried out, a clock drift measurement methodology was designed. This was required to experimentally define guard times responsible for mitigating the effect of devices' desynchronization over time. For this, we followed a workflow involving different stages. First, a GPS module was connected via serial to an external MCU, comparing, in the same machine, timestamps provided on request by the external MCU with the target RTC. Second, the measurement setup was optimized reducing the number of serial connections by synchronizing the machine via NTP (Network Time Protocol) and omitting the GPS. However, several inconsistencies were found in both measurement setups as a result of excessive serial connections. Finally, we devised the third and definite setup, which consisted of monitoring target devices toggling a GPIO pin at every slot boundary (further details are provided in Section V-A).

The next hurdle to be overcome was related to synchronization. Since LoRaWAN end devices operate under Class

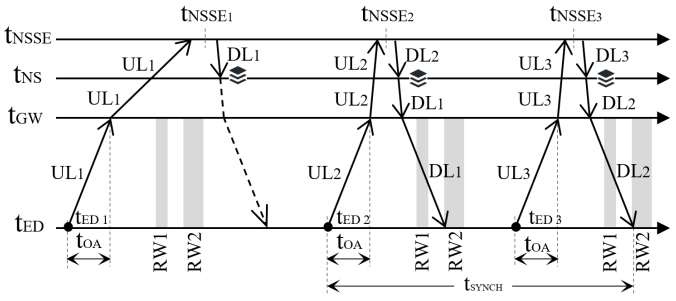


Fig. 3. Asymmetric network delays buffering packets at the network server and alternative synchronization mechanism.

A specification, the delivery of synchronization replies is conditioned by the specific opening times of the two receive windows. Thus, an occasional network delay can prevent the network server from reaching any of the windows, which will be queued and pushed after the next uplink frame received. To overcome this issue, request and reply identifiers were included in MAC payloads. As a result, devices are responsible for mapping request timestamps with their corresponding replies, discarding replies generated during high latency.

The worst-case scenario is shown in Fig. 3, where a network delay occurs when forwarding the synchronization request UL_1. The network server is not able to deliver DL_1 in any of the receive windows, and is then buffered and sent upon the next uplink, UL_2. However, the end device will discard DL_1 and schedule an additional request, UL_3, aimed at receiving the NSSE synchronization parameters corresponding to UL_2 (when no unexpected network delays occurred). Although this method would require a minimum of two synchronization retries, it is important to note that this scenario is unlikely to happen. Nevertheless, all possible scenarios needed to be considered to ensure the desired synchronization performance.

Finally, the Bloom filter hashing technique needed to be implemented in both ends (NSSE and LoRaWAN devices) to enable scheduling. The implementation was based on the non-cryptographic hash function Murmur3 [35], using a filter size of 64 bits and 6 hashing functions. Since the available implementations (x86 and x64) are optimized for their respective platforms with different output values, the x86 version was used with minor modifications being made at the end-device side to fit with fixed-point arithmetic constraints.

Divide operations were performed using an open-source library for optimized integer division [36], which enabled us to reduce the execution time and avoid overflow when performing 64-bit by 32-bit divisions. To do this, all 64-bit hashes were split into 32-bit words and divisions were only based on multiply or shift operations, with latency being up to 30 times lower than that of integer division.

V. Results and Discussion

A. Experimental setup

The measurements were carried out using a logic analyzer, the Saleae Logic Pro 16, which monitored in real-time toggling GPIOs triggered at every end-device slot boundary. Specifically, 4 pins of each device were monitored simultaneously during scheduling measurements (16 channels) – *slot pin* (toggles on slot completion), *guard pin* (set to high during the guard time), *tx pin* (set to high during the time on air) and *synch pin* (set to high during the synch margin) – while only 4 where monitored for clock-skew and synchronization-accuracy measurements. That resulted in sampling rates of 100 ns/s and 400 ns/s, respectively, whose impact on the experiments outcomes are measurement errors of ± 5 ns in scheduling and ± 1.25 ns in clock-drift and synchronization results (detailed hereafter). The channel-to-channel skew introduced by the analyzer was, at maximum, of 1 sample. These measurement errors were, in all cases, more than five orders of magnitude smaller than the magnitude being sensed. The measurement time was 60 minutes, after which the output of the logic analyzer was parsed offline to compute the time drifted at devices with respect to the reference time. This offset can be positive (meaning that a device is behind schedule) or negative (the device is ahead of schedule).

The total number of RTC alarms triggered per unit of time was found to influence the perceived time drift during the measurements. As a result, we designed a preliminary experiment to mitigate external sources of error and reach separate results regarding: (i) the added error per RTC alarm triggered, and (ii) the frequency drift of devices. To do so, 16 measurements were carried out per device varying the number of alarm triggers during a fixed time of 60 minutes. The linear regression results were computed according to (7):

$$t_i(t_0) = E_{rtc_i} \cdot n_{rtc} + C_i(t_0) \tag{7}$$

where t_0 is the measurement time reference, E_{rtc_i} is the unitary error per RTC alarm, n_{rtc} is the number of alarms triggered, and C_i is the total time drifted.

To vary the total number of RTC alarms triggered, we tested four different uplink periods and numbers of slots per end device (by setting different slot lengths). The longer the uplink period, the more alarms were triggered due to transmission slots being split into three time lengths. Likewise, the shorter the slots, the more slot-boundary alarms there were per time unit. Table I shows the configurations tested.

TABLE I
RTC ALARMS TRIGGERED FOR DIFFERENT UPLINK PERIODS AND TIME SLOT LENGTHS (MEASUREMENT TIME IS 60 MINUTES).

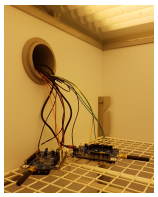
	T_{UL} = none	T_{UL} = 50 s	T_{UL} = 30 s	T_{UL} = 10 s
TS_{len} = 0.25 s	14400	14544	14640	15120
TS_{len} = 0.50 s	7200	7344	7440	7920
TS_{len} = 1.00 s	3600	3744	3840	4320
TS_{len} = 2.00 s	1800	1944	2040	2520

After computing the linear regression results, the RTC error was compensated and the time drifted by devices over the measurement time was then obtained. Once compensated, two sets of experiments were defined to obtain:

1) The clock skew of each device at room temperature: 10 measurements were carried out at approximately 20ºC and 60% of relative humidity (RH) calculating the maximum, minimum, mean and standard deviation (SD) of each clock skew. These results were used then to design the guard times in the scheduling measurements.

(a) Fitotron SGC097 chamber.

(b) Outside.



(c) Inside.

Fig. 4. Setup of temperature-controlled measurements: (a) chamber, (b) logic analyzer and laptop, (c) LoRaWAN devices.

TABLE II
LoRaWAN TIME-ON-AIR BREAKDOWN (20-BYTE PAYLOADS) AND UPLINK PERIOD OF END DEVICES IN THE NETWORK.

SF	t_{symb} [ms]	$t_{preamble}$ [ms]	t_{OA} [ms]	T_{UL} [s]
SF7	1.024	12.544	56.576	6
SF8	2.048	25.088	102.912	10
SF9	4.096	50.176	185.344	19
SF10	8.192	100.352	370.688	37
SF11	16.384	200.704	741.376	74
SF12	32.768	401.408	1318.912	132

2) The clock drift of two of the devices in a temperature-controlled environment: these devices (A and B) were placed inside a temperature-controlled chamber (Fitotron SGC097) to measure the clock drift of each one for different temperature conditions. Fig. 4 shows the equipment used and setup of devices, which remained isolated inside the chamber for five consecutive hours at temperatures ranging from 10°C to 30°C.

During all measurements, pressure was in the range of 935 to 945 hPa, and devices were parallel to the ground plane.

Having measured the clock skew of devices at room temperature, the required time on air was calculated according to [3], using a MAC payload size of 20 bytes (of which 7 bytes were reserved as the frame payload) and the following LoRaWAN configuration: $t_{preamble}$ was 8 symbols; header was enabled (H equal to 0); low data rate optimization, DE, was enabled for SF11 and SF12; CRC was activated; and the CR was set to 4/5. The resulting time on air for each SF is shown in Table II, where the preamble duration is included for further discussion. Based on these times, the uplink period of devices was minimized for each SF, using a duty cycle of 1%, also shown in Table II. In order to achieve an equivalent air-time usage in unsynchronized scenarios without increasing the number of devices, the uplink period was set to $4 \cdot TS_{len}$. Further discussion is provided in Section V-C.

To measure the synchronization accuracy of the end-to-end implementation, a set of 20 measurements was carried out for each SF and pair of devices (120 measurements per pair of devices, that is, 720 samples in total). The methodology consisted of using the logic analyzer to monitor the slot boundaries of the four devices right after synchronization. The time shift between each pair of devices was measured and synchronization-accuracy statistics were then computed.

Finally, the time slot lengths were calculated based on the measured clock skew of each device, the time on air, and the mean synchronization accuracy. Taking these slot lengths as reference, additional testing scenarios were proposed to evaluate the performance of the scheduling proposal. Table III shows all the scenarios tested, summarized as follows:

- Unsynchronized (slotted): time is divided in unsynchronized slots. Note that, despite not being included in Table III, a pure Aloha behavior was also tested. In both scenarios, the uplink period of devices was computed as $4 \cdot TS_{len}$, with 3 seconds being the lower threshold due to Class A specification limitations. This results in 3-second periods for SF7-10, and approximately 3.5-second and 6-second ones for SF11 and SF12 respectively.
- Synchronized (TS_0): this scenario represents the baseline version of the scheduling proposal, where: the guard time (t_{guard}) is computed as twice the worst-case relative drift in the network, that is, twice the difference between the fastest and the slowest clock; the synchronization margin (t_{synch}) is set to the maximum of its mean values. Synchronization is performed in-band and transmission slots are allocated adjacently based on each device's EUI.
- Synchronized (TS_{1-4}): uses the baseline configuration, modifying the guard time (t_{guard}) and synchronization margin (t_{synch}): TS_1 halves the guard time; TS_2 bases its guard time on the two most-similar clocks; TS_3 removes the guard time; and TS_4 removes the guard time and sets to the mean-of-means the synchronization margin.
- Synchronized (Bloom filter): baseline time slot lengths are used, switching the slot-allocation mechanism to Bloom filter probabilistic data structures. Hence, slots will not necessarily be allocated adjacently and the false-positive probability in Eq. (2) plays an important role.
- Synchronized (out of band): baseline time slot lengths are used, increasing the resynchronization duty cycle to 10%.

To give an idea about MCU requirements towards implementation, the RAM and Flash memory usages for the baseline scheduling setup were of 11.344 KB and 86.156 KB while, for the unsynchronized, of 10.280 KB and 85.064 KB. That is, the synchronization overhead in terms of RAM and Flash memory occupation was of 5.32% and 0.57%, respectively.

The measurement time was 90 minutes in the scheduling experiments, with 15-minute resynchronization periods. Thus, guard time was defined as twice the time difference between the fastest and the slowest clock after 15 minutes. Uplink periods were maximized to achieve high network loads. In the Aloha scenario, every new transmission was randomized while, in the rest, randomization happened only before the first transmission. Specifically, a random delay of up to 3 minutes was added prior to sending synchronization requests. Hence, repeated measurements were carried out in the unsynchronized slotted scenario to widen the confidence interval.

In the current implementation, slots can be assigned in parallel taking advantage of orthogonality. Nevertheless, single-

TABLE III
SCHEDULING SCENARIOS ACCORDING TO TIME SLOT LENGTHS (TSL) AND SYNCHRONIZATION SETUPS (SYNCHRONIZATION BAND AND SLOTS-ALLOCATION TECHNIQUE).

Scenario		Unsynch (slotted)	Synch (TS_0)	Synch (TS_1)	Synch (TS_2)	Synch (TS_3)	Synch (TS_4)	Synch (Bloom)	Synch (OOB)
Synch band		–	In band	In band	In band	In band	In band	In band	Out of band
Slot allocation		–	Fixed	Fixed	Fixed	Fixed	Fixed	Bloom filter	Fixed
Synch duty cycle [%]		–	0.1	0.1	0.1	0.1	0.1	0.1	10
t_{guard} [ms]		145	145	72.5	3	0	0	145	145
t_{synch} [ms]		16	16	16	16	16	10	16	16
TS_{len} [ms]	SF7	218	218	146	77	73	67	218	218
	SF8	264	264	191	122	119	112	264	264
	SF9	346	346	274	204	201	195	346	346
	SF10	532	532	459	390	387	361	532	532
	SF11	902	902	830	760	757	751	902	902
	SF12	1480	1480	1407	1338	1335	1329	1480	1480

TABLE IV
LINEAR REGRESSION RESULTS AND GOODNESS-OF-FIT MEASUREMENT SHOWING THE ERROR ADDED PER RTC ALARM AND CLOCK SKEW.

Device	E_{rtc_i} [ms]	C_i [ms]	R^2
A	0.122011	-375.129614	0.999990
B	0.122033	-91.117528	0.999992
C	0.122026	-360.174126	0.999984
D	0.122049	-84.829399	0.999994

TABLE V
CLOCK SKEW OF END DEVICES.

Device	$\overline{\Delta t_{i,1h}}$ [ms]	Max [ms]	Min [ms]	SD [ms]	$\overline{\Delta f_i}$ [ppm]
A	-376.532	-374.948	-378.21	0.989	105
B	-92.297	-91.457	-93.699	0.744	26
C	-361.292	-360.439	-362.608	0.691	100
D	-86.525	-85.018	-87.283	0.906	24

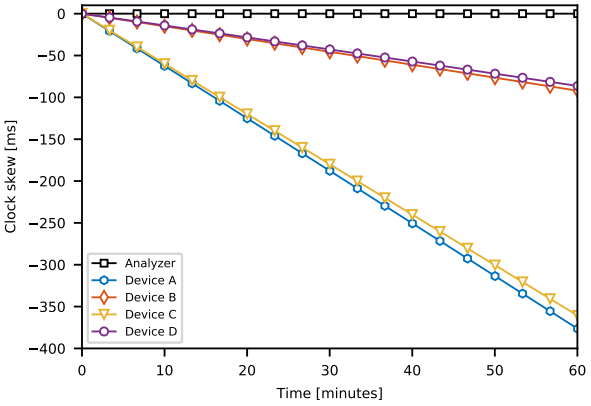


Fig. 5. Clock skew of the devices over time.

SF setups were defined to reach independent conclusions about guard time influence on packet losses. To enable a fairer comparison, the two unsynchronized scenarios were tested as well under a single-SF approach. Otherwise, packet delivery ratios (PDRs) achieved would increase significantly.

B. Clock drift measurement

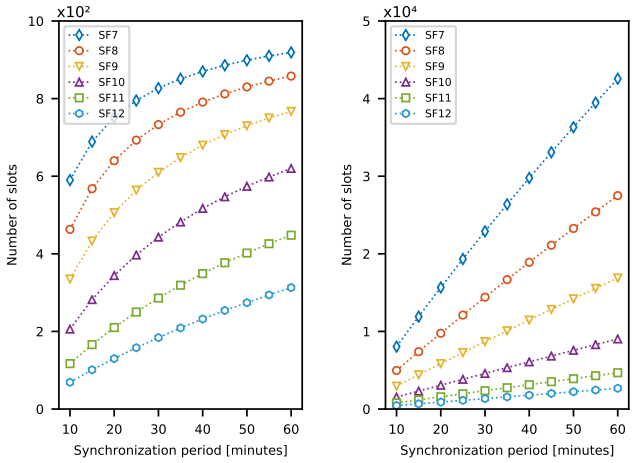
Table IV shows the linear regression results from the preliminary clock-drift test, in addition to their goodness-of-fit measure. This was intended to determine the time drifted by devices due to the total number of RTC alarms triggered and time elapsed. The analysis was based on (7), with the dependent variable being the unitary error committed per RTC alarm, E_{rtc_i} and, the independent, the clock skew of devices in milliseconds over a measurement time of 60 minutes (C_i).

As highlighted during the implementation, the end devices operate in RTC ticks to mitigate conversion errors from milliseconds to ticks due to their fixed-point arithmetic and 32-bit architecture. Hence, in view of the results, every new RTC alarm trigger added an approximated delay of 0.122 milliseconds, that is, 1 tick under 8192 ticks-per-second resolution. If not compensated, the perceived time drifted by a device can increase considerably during long measurement periods. For instance, in SF7 scenarios with 218-millisecond slot lengths and 6-second uplink periods, the accumulated error after 60 minutes rises to approximately 1.8 seconds. Therefore, this error was compensated at the end-device side on each RTC alarm trigger, influencing all the results provided hereafter.

After RTC error compensation, 10 measurements were carried out at room temperature to measure the clock skew of the concerned devices (see Table V). The results show the four development boards drift in the negative time direction (ahead of schedule). This means that their clocks oscillate faster than their nominal frequency, with Device A having the highest clock skew (approximately 375 milliseconds are lost after 60 minutes). Fig. 5 provides a detail of one of the ten clock-skew measurements performed, where the cumulative time drifted by the four end devices is shown over the 60-minute measurement period. The four clock skews are represented with respect to the logic analyzer time reference.

It is important to note that the two pairs of devices used (A–C and B–D) have similar clock skews (105–100 ppm and 26–24 ppm, respectively) which, as highlighted in Section IV, belong to two different manufacturing series. This explains the difference in the clock skew of the devices over time, in contrast to the information provided by the IC manufacturer ($\approx$ 20 ppm). This fact corroborates the importance of measuring the clock skew experimentally as part of the validation of the scheduling proposal.

Based on these results, the guard time of devices was defined in order to proceed with synchronization and scheduling measurements. For this, considering a 15-minute resynchronization period ($t_0 = 15$), the guard time was computed as $2 \cdot (C_A(t_0) - C_D(t_0))$, that is, 145 milliseconds (see Table III). The four devices used for the implementation having



(a) Heterogeneous network. (b) Homogeneous network.

Fig. 6. Maximum number of assignable slots per synchronization period for (a) heterogeneous networks (clocks A-D) and (b) homogeneous (clocks B-D).

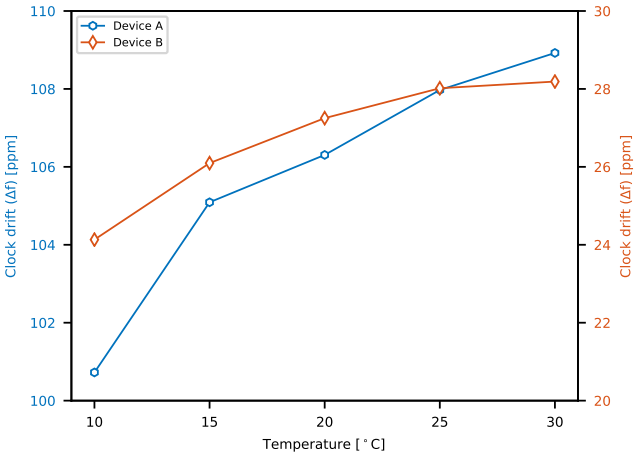


Fig. 7. Steady-state clock drift vs. temperature.

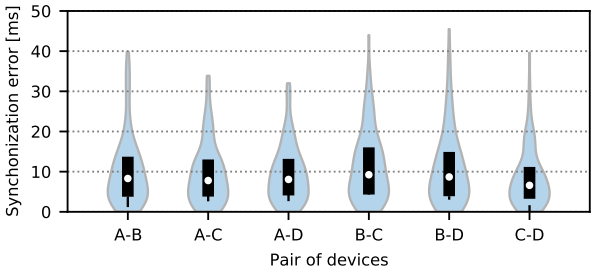


Fig. 8. Synchronization error for each pair of devices.

noticeable differences in clock skew underlines an important design aspect of the implementation, since the definition of slot lengths throughout the network is conditioned by the difference between the fastest and slowest clock.

Based on the four clocks analyzed, Fig. 6 measures the impact of heterogeneous clock sources on the maximum number of assignable slots per synchronization period. The heterogeneous scenario considers the coexistence of the fastest and slowest clocks (Devices A and D) while the homogeneous considers the two lowest-skew devices (B and D). As can be seen, there is a trade-off between synchronization overhead and number of available slots: the longer the synchronization period (lower overhead), the less available slots per time unit due longer guard-time definitions. This shows the consequences of moving from a simulator to a real implementation where, even when using a few devices with common specifications, heterogeneous clocks are found. In single-gateway networks, an additional trade-off arises: the higher the synchronization period, the less devices that can be served due to duty cycle limitations. Co-located gateways, however, will alleviate this effect. The optimal synchronization period is determined by resolving the mathematical optimality problem presented by Haxhibeqiri *et al.* [7].

Finally, the clock drift of Devices A and B was measured in a temperature-controlled chamber. Fig. 7 shows these results, measured after a 4-hour stabilization period of all the devices inside the chamber. Despite the temperature coefficients of the quartz crystals being unknown, the influence of different temperatures on each of the clocks used is noticeable.

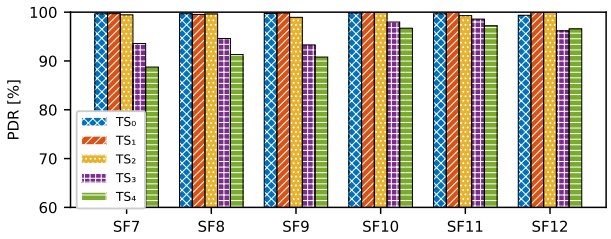
According to the parabolic frequency dependence of crystals over temperature (with negative temperature coefficients), the frequency of devices is expected to slow down at temperatures below the turnover point, as shown in (1). Furthermore, a higher RH (at lower temperatures) is also responsible for increasing the capacitance of the oscillator circuit and reducing the oscillation frequency [37].

The results confirm these hypotheses, but it is important to note how the two boards present different responses to the same environmental changes. Device B not only has lower skew, but also lower sensitivity to drastic temperature changes. Conversely, Device A presented greater dependency on temperature, with several instabilities being observable. This highlights the importance of clock drift measurement according to the end application pursued, which can significantly influence the behavior of devices. Please note that the clock drift results cannot be directly compared to those at room temperature, since these were measured at different RH levels.

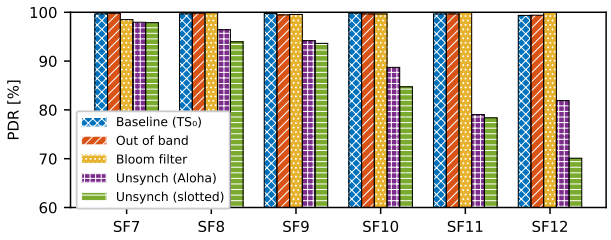
C. LoRaWAN synchronization and scheduling

The distribution of the results for synchronization accuracy is provided in Fig. 8, in the form of violin plots. The measurements were carried out for each pair of devices at different SFs but, since no significant difference was found while synchronizing over different SFs, each column merges 20 synchronization-error results per SF, that is, 120 measurements. With a sample size of 720, mean of 10.002 milliseconds and a SD of 8.202 milliseconds, the error committed in the synchronization-accuracy measurements was lower than 0.787 milliseconds (using a confidence interval of 99%).

Two accuracy indicators were used to define time slot lengths in the scheduling measurements: the maximum arithmetic mean of synchronization errors for each pair of devices (worst case) and the arithmetic mean of synchronization-error means (overall mean). As a result, t_{synch} in (5) was set to 16 or 10 milliseconds, respectively, depending on the scenario setup. Please see Table III for further information.



(a) PDR achieved using different time slot lengths (baseline setup is TS_0).



(b) PDR achieved under different setups (synch vs. unsynch scenarios).

Fig. 9. PDR for different setups.

Finally, the end-to-end evaluation was carried out testing all the scenarios defined in Table III. Fig 9a shows PDRs using variable slot lengths (scenarios TS_{0-4}). Despite TS_1 consisting of half the guard time used in the baseline slot length (TS_0), both led to a similar ratio of packets delivered for all SFs. The high PDR achieved for TS_1 can be explained according to (6), where the maximum offset between transmissions assigned to neighbor slots is quantified assuming the same drifting direction. In our setup, however, δ was lower than 3 minutes (given by the randomized delay performed by devices prior to the first synchronization). Thus, for most of the measurement time ($T_{synch} - t_\delta$), the clock offset between the fastest and the slowest clocks remained lower than their relative drift, with the t_{guard} defined in Table III being sufficient to prevent overlapped transmissions. As a result, the ratio of delivered packets was nearly 100%.

Likewise, the two devices with the most similar clocks (B and D) achieved a successful PDR in the TS_2 scenario, even with the baseline guard time being reduced from 145 to 3 ms. Devices were assigned to adjacent slots, with Device D being allocated in the first position to force its overlap with Device B over time (worst-case scenario). The results confirm that, in the case of having deployments based on similar clock sources, the maximum number of assignable slots can be considerably increased (see Fig. 6) without hindering the reliability of the scheduling technique. Furthermore, the higher density of slots in this setup validates the suitability of the measured slot-length fields (clock skew of devices and synchronization accuracy).

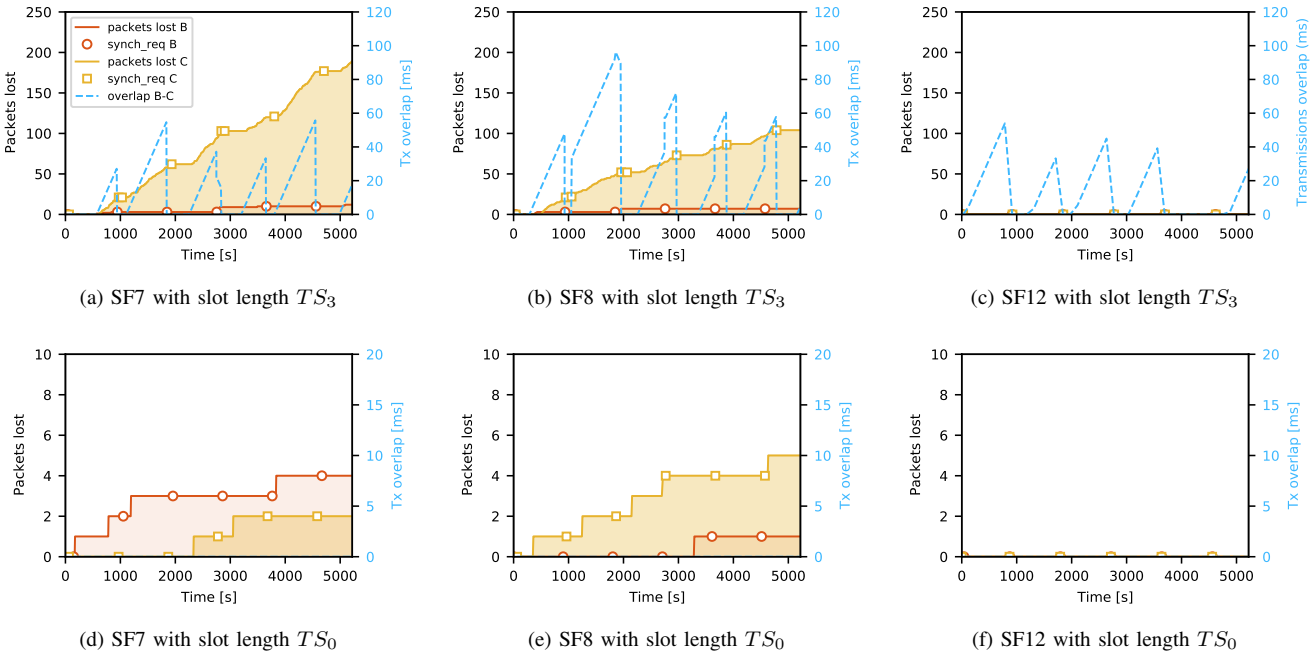
In contrast, PDRs for setups TS_3 and TS_4 were considerably influenced by the absence of guard time. However, the impact of using slightly different slot lengths is negligible compared to that of a variable synchronization error (see Fig. 8). Analyzing the results of both setups per SF, Fig. 9a shows how the removal of guard time has a greater influence at lower SFs. This is consistent with previous studies, such as [38] and [9], where the authors evaluated the impact of preamble symbols interfered with the successful reception of a LoRa packet. For further discussion, Fig. 10 provides a detailed comparison of the TS_0 and TS_3 setups, with the cumulative number of packets lost being monitored (left-hand axis) simultaneously with the overlap of transmissions in ms (right-hand axis). The synchronization events of each device are represented with markers, which show how every new synchronization re-aligns slot boundaries preventing overlap.

Since no guard time is present in the TS_3 scenarios (Figs. 10a to 10c), the clock skew of devices B (slower) and C (faster) provokes the misalignment of slot boundaries over time. The maximum theoretical overlap prior to resynchronization is around 75 milliseconds, according to (6), but is influenced by the synchronization error committed. Thus, the lower the SF, the sooner the losses start, due to shorter preamble lengths. This is especially harmful for SF7 and SF8 (Fig. 10), where preamble lengths are lower than the maximum overlap (see preamble-length definitions in Table VI). Conversely, Figs. 10d to 10f show how the scheduling scheme with the baseline slot-length definitions prevent LoRaWAN devices from colliding, where practically no failed transmissions are found using the highest-possible uplink periods under a 1% duty cycle.

Regarding PDR for a baseline slot length and different setups (see Fig.9b), no noticeable difference is found while synchronizing in-band (baseline scenario) versus out-of-band. When fixed-slot allocation is performed, the NSSE appends a set of forbidden slots to the synchronization reply so that devices skip their periodicity so as not to interfere with scheduled resynchronization requests. Thus, an improvement in PDR will only be noticed for unassigned synchronization requests, such as the first synchronization event or a synchronization retry, which are negligible with respect to the total number of packets transmitted (160 when using SF12 to 3500 using SF7). In fact, the main difference between the in-band and out-band results lies in this total number of transmitted packets. While the 10-% duty cycle in out-band synchronization permits constant traffic during the entire measurement time, in-band synchronizes under 0.1-% duty cycle, provoking periods of inactivity of the devices with failed synchronizations.

Likewise, PDRs achieved using Bloom filters are comparable to those of the baseline scenario. Considering that, in the former, end devices are not necessarily assigned to consecutive slots, the chances of collisions caused by overlapped transmissions are low. This is observed for SF8 to SF12. However, the high uplink periodicity at lower SF (6 seconds) maximizes the false-positive probability in (2) forcing some simultaneous transmissions. This can be seen at SF7. It is important to note that, despite the Bloom-filter implementation being aimed at infrequent transmissions, traffic was maximized in our setups to test the worst-case traffic load scenarios.

PDR results for unsynchronized scenarios cannot be directly compared to those of the synchronized ones, since the unitary air-time usage differs in both cases. For instance, our baseline scenario allocates end devices to neighbor slots (fixed-slot allocation) with air-time usages ranging from 26% to 89% on a slot basis. These are even higher if lower guard times are considered (as in scenarios TS_1 and TS_3). Table VI compares the PDR results achieved for synchronized scenarios



(a) SF7 with slot length TS_3 (b) SF8 with slot length TS_3 (c) SF12 with slot length TS_3

(d) SF7 with slot length TS_0 (e) SF8 with slot length TS_0 (f) SF12 with slot length TS_0

Fig. 10. Packets lost (left-hand axis) and overlap of transmissions (right-hand axis) over time, for TS_0 and TS_3 setups.

TABLE VI
TIME ON AIR CONSUMED PER TIME UNIT AND PDR RESULTS FOR UNSYNCHRONIZED AND SYNCHRONIZED SCENARIOS.

SF	Unsynch slotted		Synch scenarios		
	t_{OA} [%]	PDR [%]	t_{guard} [ms]	t_{OA} [%]	PDR [%]
SF7	8	97.9	0	78	93.6
			72.5	39	99.7
			145	26	99.7
SF8	13	94.0	0	87	94.6
			72.5	54	99.6
			145	39	99.7
SF9	25	93.4	0	92	93.3
			72.5	68	99.9
			145	54	99.7
SF10	49	84.7	0	96	98.0
			72.5	81	100
			145	70	99.8
SF11	82	78.4	0	98	98.6
			72.5	89	100
			145	82	99.6
SF12	89	70.1	0	99	99.4
			72.5	94	100
			145	89	99.3

(TS_0, TS_1 and TS_3) with those of unsynchronized scenarios, including the air-time usage per setup.

In unsynchronized setups, the uplink period was increased to $4 \cdot TS_{len}$ to reach the same air-time usages as in synchronized setups, on a slot basis. However, this was only possible for SF11-12, since the resulting uplink period for the remaining SFs was lower than the minimum threshold imposed by the LoRaWAN Class A specification (3 seconds). This is, in fact, one of the practical limitations of our setup, where a minimum of 14 end devices would have been required to deploy an unsynchronized scenario with traffic conditions equivalent to that of the baseline. Nevertheless, even with higher air-time percentages, the synchronized setup outperforms the unsynchronized one. For unsynchronized SF7 to SF10, despite having much lower air-time usages (8% to 49%) than those of their equivalent synchronized scenarios (26% to 70%), the PDRs were up to 16% worse. This is even more significant for SF11 and SF12, where both scenarios had the same air-time occupancy per SF. In this case, the scheduling technique led to PDR improvements of 21.2% and 29.2%, respectively.

VI. CONCLUSION

This work describes the methodology and challenges overcome towards implementing and evaluating a low-overhead scheduling technique on top of LoRaWAN Class A devices, where the NSSE receives synchronization requests from end devices and assigns slots based on uplink periodicities. Since the technique was previously designed but only validated in a simulation environment, our major contribution lies in the experimental deployment and validation of an operational end-to-end system. This work fills a literature gap by providing a methodology design and a real-world approach to clock drift measurement, LoRaWAN synchronization and scheduling.

During implementation, several hurdles were overcome and drawn together as a set of lessons learned. On the end-device side, challenges overcome include: slot-transition mechanism, clock source selection, error compensation, granularity, and 32-bit adaption of Bloom filter hashing for fixed-point constraints. Moreover, two sources of error were identified and compensated: alarm-triggering delay on a slot basis, and millisecond-to-tick accumulated conversion errors. Regarding communication, a priority mechanism for synchronization traffic was implemented based on LoRaWAN frame ports, while buffering issues at the network server caused by asymmetric network delays were also overcome. The result was an operational end-to-end system.

A set of measurements was conducted achieving a mean synchronization accuracy of 10 milliseconds, after which the overall concept was validated. Our results, first, highlight the benefits of such a scheduling entity, especially in high network loads. PDR improvements of up to 29.2% were achieved with respect to an analogous unsynchronized setup. Second, the clock-drift measurement methodology is validated, with guard times defined preventing an overlap of adjacent transmissions. Nevertheless, even when guard times in the network are removed, synchronization of end devices enables considerably higher PDRs than those of an unsynchronized scenario.

This work provides practical insights and findings on LoRa-based scheduling implementation. Even when using a few devices with the same technical specifications, the noteworthy differences found in their clock skew were key to define time slot lengths. This implied the definition of longer-than-expected guard times and, thus, a reduction of synchronization periods with respect to those that were defined in the previous simulation study. This practical insight might serve as a benchmark to define more realistic scenarios in future related works. These lessons learned are of interest to researchers conducting alternative real-world LoRa scheduling schemes, since most of the barriers encountered were imposed by the constrained nature of hardware and the LoRaWAN MAC behavior of Class A. In addition, the feasibility of the scheduling concept was demonstrated, resulting in an operational end-to-end system able to synchronize LoRaWAN end devices over time and, hence, enable larger-scale deployments with occasional collisions. The achieved delivery ratios were close to 100%, with air-time usages up to 89% and 99% (SF12), depending on whether heterogeneous or homogeneous clock sources were used. Such a LoRa scheduling scheme could be adopted in a wide range of application domains to alleviate existing scalability concerns. That is the case of asset tracking and inventory monitoring in the logistics industry, where numerous battery-powered devices are expected to transmit sensor data periodically that needs to be accordingly scheduled.

To further improve the algorithm, we plan to focus on the flexibility with which different nodes can start (or stop) being part of the network, which might have different resynchronization needs, tx configurations or payload lengths according to their goal. In the current version, time-slot lengths are assigned statically based on the worst clock. The lack of a more dynamic scheme to measure their clock skew and allocate slots in the network could become a barrier against its adoption in scenarios consisting of mobile nodes which, in fact, are likely to be involved in the aforementioned use case. This feature will be improved in the future to conduct a real-world deployment in the asset-tracking domain, with all the implications this entails. For this, a preliminary study of the influence of clock sources on the MCU, RTC and transceiver would be highly desirable. Moreover, with a view to reduce the effect caused by a limited amount of downlink resources in single-gateway networks, the proposed scheme will be extended to enable synchronization based on beacons (following the Class B specification). Both approaches will then be compared experimentally in terms of total energy overhead, synchronization accuracy, and network capacity.

ACKNOWLEDGMENT

This work was partially supported by the Flemish FWO SBO S004017N IDEAL-IoT (Intelligent DEnse And Longe range IoT networks) project and by Spanish Ministerio de Ciencia e Innovación (FEDER, EU funds) under RTI2018-098156-B-C52 and ECO2016-75781-P projects. Celia Garrido-Hidalgo holds 2019-PREDUCLM-10703 fellowship from FEDER, EU, at Universidad de Castilla-La Mancha (UCLM). We thank the Hydrogeology Group from UCLM for providing SCG097 equipment.

REFERENCES

[1] K. L. Lueth, "State of the IoT & Short-term Outlook-Q1/Q2 2018," IoT Analytics GmbH, Hamburg, Germany, Tech. Rep., 2018.
[2] *LoRaWAN™ 1.0.3 Specification*, LoRa Alliance, 6 2018, rev. 1.0.3.
[3] *SX1272/3/6/7/8: LoRa Modem*, Semtech, 7 2013, rev. 1.
[4] U. Raza, P. Kulkarni, and M. Sooriyabandara, "Low power wide area networks: An overview," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 855–873, 2017.
[5] L. Chasserat, N. Accettura, and P. Berthou, "Short: Achieving energy efficiency in dense LoRaWANs through TDMA," in *IEEE Int. Symp. a World of Wireless, Mobile and Multimedia Netw. (WoWMoM)*, 2020.
[6] H. Rajab, T. Cinkler, and T. Bouguera, "IoT scheduling for higher throughput and lower transmission power," *Wireless Netw.*, pp. 1–14, 2020.
[7] J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Low Overhead Scheduling of LoRa Transmissions for Improved Scalability," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3097–3109, 2019.
[8] O. Seller and N. Sornin, "Low power long range transmitter," *US Patent 20140219329 A*, vol. 1, p. 2014, 2014.
[9] 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, 2017.
[10] 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, p. 3995, 2018.
[11] D. Magrin, M. Capuzzo, and A. Zanella, "A thorough study of LoRaWAN performance under different parameter settings," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 116–127, 2020.
[12] *Short Range Devices (SRD) operating in the frequency range 25 MHz to 1 000 MHz*, European Telecommunications Standards Institute (ETSI), 2 2017, rev. 3.1.1.
[13] F. Van den Abeele, J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Scalability analysis of large-scale LoRaWAN networks in ns-3," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2186–2198, 2017.
[14] B. Reynders, Q. Wang, P. Tuset-Peiro, X. Vilajosana, and S. Pollin, "Improving reliability and scalability of lorawans through lightweight scheduling," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1830–1842, 2018.
[15] K. Q. Abdelfadeel, D. Zorbas, V. Cionca, and D. Pesch, "$FREE$—Fine-Grained Scheduling for Reliable and Energy-Efficient Data Collection in LoRaWAN," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 669–683, 2020.
[16] R. K. Singh, M. Aernouts, M. De Meyer, M. Weyn, and R. Berkvens, "Leveraging LoRaWAN Technology for Precision Agriculture in Greenhouses," *Sensors*, vol. 20, no. 7, p. 1827, 2020.
[17] G. Tresca, F. Vista, and P. Boccadoro, "Experimenting LoRa-compliant solutions in Real-World Scenarios," *Internet Technology Letters*, vol. 3, no. 2, p. e136, 2020.
[18] C. Garrido-Hidalgo, T. Olivares, F. J. Ramirez, and L. Roda-Sanchez, "An end-to-end Internet of Things solution for reverse supply chain management in Industry 4.0," *Computers in Industry*, vol. 112, p. 103127, 2019.
[19] P. Boccadoro, B. Montaruli, and L. A. Grieco, "Quakesense, a LoRa-compliant earthquake monitoring open system," in *2019 IEEE/ACM 23rd International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*. IEEE, 2019, pp. 1–8.
[20] D. Zorbas, K. Abdelfadeel, P. Kotzanikolaou, and D. Pesch, "TS-LoRa: Time-slotted LoRaWAN for the Industrial Internet of Things," *Comput. Commun.*, vol. 153, pp. 1–10, 2020.
[21] R. Piyare, A. L. Murphy, M. Magno, and L. Benini, "On-demand LoRa: Asynchronous TDMA for energy efficient and low latency communication in IoT," *Sensors*, vol. 18, no. 11, p. 3718, 2018.

[22] F. Tirado-Andrés and A. Araujo, "Performance of clock sources and their influence on time synchronization in wireless sensor networks," *Int. J. Distrib. Sens. Netw.*, vol. 15, no. 9, p. 1550147719879372, 2019.
[23] M. Lévesque and D. Tipper, "A survey of clock synchronization over packet-switched networks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2926–2947, 2016.
[24] T. Polonelli, D. Brunelli, A. Marzocchi, and L. Benini, "Slotted ALOHA on LoRaWAN: Design, Analysis, and Deployment," *Sensors*, vol. 19, no. 4, p. 838, 2019.
[25] T. Polonelli, D. Brunelli, and L. Benini, "Slotted ALOHA Overlay on LoRaWAN: A Distributed Synchronization Approach," in *2018 IEEE 16th Int. Conf. on Embedded and Ubiquitous Comput. (EUC)*. IEEE, 2018, pp. 129–132.
[26] 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.
[27] M. Rizzi, A. Depari, P. Ferrari, A. Flammini, S. Rinaldi, and E. Sisinni, "Synchronization uncertainty versus power efficiency in LoRaWAN networks," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 4, pp. 1101–1111, 2018.
[28] C. Ebi, F. Schaltegger, A. Rüst, and F. Blumensaat, "Synchronous LoRa mesh network to monitor processes in underground infrastructure," *IEEE Access*, vol. 7, pp. 57 663–57 677, 2019.
[29] Dash7 Alliance, "OSS-7: Open Source Stack," https://github.com/MOSAIC-LoPoW/dash7-ap-open-source-stack, accessed: 15-6-2020.
[30] ChirpStack, "Open Source LoRaWAN Network Server stack," https://www.chirpstack.io/, accessed: 15-6-2020.
[31] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click Modular Router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, p. 263–297, 2000.
[32] *Sub-G Module Datashheet - CMWX1ZZABZ-091*, Murata Investment Co., Ltd., 10 2018, rev. C.
[33] *Using the hardware RTC and the tamper management unit (TAMP) with STM32 microcontrollers - AN4759*, STMicroelectronics, 2 2020, rev. 5.
[34] Semtech and StackForce, "LoRaMAC 4.4.4. API documentation," https://stackforce.github.io/LoRaMac-doc/, accessed: 17-6-2020.
[35] A. Appleby, "MurmurHash3," https://github.com/aappleby/smhasher/wiki/MurmurHash3, accessed: 22-6-2020.
[36] "Libdivide – Header-only C/C++ library for optimizing integer division," https://github.com/ridiculousfish/libdivide, accessed: 22-6-2020.
[37] X. Yu, X. Chen, X. Ding, and X. Zhao, "A high-stability quartz crystal resonator humidity sensor based on tuning capacitor," *IEEE Trans. Instrum. Meas.*, vol. 67, no. 3, pp. 715–721, 2018.
[38] M. Bor, J. E. Vidler, and U. Roedig, "LoRa for the Internet of Things," in *Proc. Int. Conf. on Embedded Wireless Syst. Netw.* EWSN '16, 2016, pp. 361–366.

Celia Garrido-Hidalgo received her Bachelor's Degree in Industrial Electronics and Automation Engineering in 2015 and a Master's in Industrial Engineering in 2018, both from Universidad de Castilla-La Mancha, Spain. She is currently pursuing her Ph.D. in Advanced Computing Technologies at the Albacete Research Institute of Informatics (Spain) with the High-Performance Networks and Architectures group. Celia has received different international awards for innovation projects, and her research interests include low-power communication standards, heterogeneous networks and reverse supply chain management.

Jetmir Haxhibeqiri received the Masters degree in Engineering (information technology and computer engineering) from RWTH Aachen University, Aachen, Germany, in 2013. In 2019, he obtained a PhD in Engineering Computer Science from Ghent University with his research on flexible and scalable wireless communication solutions for industrial warehouses and logistics applications. Currently he is a post-doc researcher in the Internet Technology and Data Science Lab (IDLab) of Ghent University and imec. His current research interests include wireless communications technologies (IEEE 802.11, IEEE 802.15.4e, LoRa) and their application, IoT, wireless time sensitive networking, in-band network monitoring and wireless network management.

Bart Moons received the masters degree in Electronics and ICT from the University of Antwerp, in 2017, with specialization in ambient technology. During his studies, he had the opportunity to work on Wireless Sensor Networking in the Internet of Things as part of his master thesis. He is currently pursuing the Ph.D. degree with the Internet Technology and Data Science Lab of Ghent University and imec. His current research interests include the Internet of Things, Heterogeneous Wireless Networks, the Web of Things and standard based networking in the Internet of Things.

Jeroen Hoebeke received the Masters degree in Engineering Computer Science from Ghent University in 2002. In 2007, he obtained a PhD in Engineering Computer Science with his research on adaptive ad hoc routing and Virtual Private Ad Hoc Networks. Current, he is an associate professor in the Internet Technology and Data Science Lab of Ghent University and imec. He is conducting and coordinating research on wireless (IoT) connectivity, embedded communication stacks, deterministic wireless communication and wireless network management. He is author or co-author of more than 150 publications in international journals or conference proceedings.

Teresa Olivares is Assistant Professor with the Department of Computing Systems at Universidad de Castilla-La Mancha. She received her PhD degree in Computer Science in 2003 from the same university. She is a member of the research group High-Performance Networks and Architectures at the Albacete Research Institute of Informatics. Her main scientific research interests include Internet of Things (IoT) standards, communications and protocols, heterogeneous low power wireless sensor networks and standards, smart environments, Industry 4.0 and Reverse Logistics. She has participated in more than 40 research projects and has co-authored more than 70 research papers in journals, conferences and book chapters.

F. Javier Ramirez is Assistant Professor of Engineering Management at the School of Industrial Engineering, Universidad de Castilla-La Mancha, Albacete, Spain. Ramirez's research focuses on operations research and manufacturing engineering. He has participated in more than 120 R&D projects and has co-authored over 50 research papers in journals, specialized conferences and book chapters. He is co-inventor of 6 patents with industrial exploitation. F. Javier Ramirez is Associate Editor of Proceedings of the Institution of Mechanical Engineers Part C: Journal of Mechanical Engineering Science, and Cogent Engineering.

Antonio Fernández-Caballero received his M.Sc. in Computer Science from Universidad Politécnica de Madrid, Spain, in 1993, and his Ph.D. from the Department of Artificial Intelligence at Universidad Nacional de Educación a Distancia, Spain, in 2001. He is a Full Professor with the Department of Computer Science at Universidad de Castilla-La Mancha. He is head of the n&aIS (natural and artificial Interaction Systems) research group belonging to the Albacete Research Institute of Informatics. His research interests are Image Processing, Cognitive Vision, Mobile Robotics, Affective Computing and Intelligent Agents. Antonio Fernández-Caballero is Associate Editor of Pattern Recognition Letters, Topic Editor-in-Chief for Vision Systems of International Journal of Advanced Robotic Systems, and Specialty Chief Editor for Robot and Machine Vision of Frontiers in Robotics and AI, among other editorship tasks. He has authored more than 380 peer-reviewed papers.