Computer Communications 153 (2020) 1–10

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

TS-LoRa: Time-slotted LoRaWAN for the Industrial Internet of Things

Dimitrios Zorbas [a,*], Khaled Abdelfadeel [c], Panayiotis Kotzanikolaou [b], Dirk Pesch [c]

[a] Tyndall National Institute, University College Cork, Cork, Ireland
[b] Department of Informatics, University of Piraeus, Piraeus, Greece
[c] School of Computer Science and IT, University College Cork, Cork, Ireland

ARTICLE INFO

Keywords:
LoRa
Industrial Internet of Things
Synchronisation
Acknowledgements

ABSTRACT

Automation and data capture in manufacturing, known as Industry 4.0, requires the deployment of a large number of wireless sensor devices in industrial environments. These devices have to be connected via a reliable, low-latency, low-power and low operating-cost network. Although LoRaWAN provides a low-power and reasonable-cost network technology, its current ALOHA-based MAC protocol limits its scalability and reliability. A common practise in wireless networks is to solve this issue and improve scalability through the use of time-slotted communications. However, any time-slotted approach comes with overheads to compute and disseminate the transmission schedule in addition to ensuring global time synchronisation. Affording these overheads is not straight forward with LoRaWAN restrictions on radio duty-cycle and downlink availability. Therefore, in this work, we propose TS-LoRa, an approach that tackles these overheads by allowing devices to self-organise and determine their slot positions in a frame autonomously. In addition to that, only one dedicated slot in each frame is used to ensure global synchronisation and handle acknowledgements. Our experimental results with 25 nodes show that TS-LoRa can achieve more than 99% packet delivery ratio even for the most distant nodes. Moreover, our simulations with a higher number of nodes revealed that TS-LoRa exhibits a lower energy consumption than the confirmable version of LoRaWAN while not compromising the packet delivery ratio.

1. Introduction

The drive towards the Industry 4.0 vision is to reduce operational costs by automating a large number of processes by involving (mobile) robots, sensors, actuators, edge/cloud control, and autonomous vehicles [1]. The orchestration of services provided by such devices requires reliable networks that can provide very high packet delivery ratio, low latency, and in some cases ultra-low power consumption.

Current Industrial IoT (IIoT) communications involve protocols such as Wireless HART [2], IEEE 802.1 Time Sensitive Networking [3] or 6TiSCH [4]. However, some of these protocols are wired and, thus, they cannot support mobility, in addition to a high installation cost. Current IEEE802.15.4-based IIoT protocols can only achieve a short to medium radio range, therefore limiting the degree of mobility, while some of them also exhibit high latency due to the multi-hop nature of their deployments. Furthermore, these wireless standards operate in the 2.4 GHz ISM band, which is already very crowded with other wireless technologies such as IEEE802.11/WiFi often creating significant interference.

In contrast with the current IIoT protocols, a long-range technology such as LoRa, which can have a range of many kilometres under Line-Of-Sight (LOS), can mitigate some of these limitations. LoRa [5] is a proprietary spread spectrum modulation technique developed by Semtech which trades data rate with sensitivity using multiple (almost orthogonal) Spreading Factors (SF). The higher a SF, the higher the sensitivity and, thus, the longer the transmission range but the lower the data rate. The LoRa Alliance, a non-profit association consisting of Semtech as well as other companies and universities from across the world, have proposed an open standard, called LoRaWAN, to support bi-directional communication, end-to-end security, mobility, and localisation services [6].

However, LoRaWAN's development has been focused on IoT applications and on battery longevity, neglecting collision avoidance mechanisms. As a consequence, the ALOHA-based MAC-layer cannot guarantee typical IIoT requirements such as a higher than 99% packet delivery ratio and a guaranteed low delay. Moreover, a drawback of LoRaWAN is that it operates in the unlicensed spectrum in which strict radio duty cycle regulations are applied for most of the bands [7]. This restriction sets a lower bound on the time between successive transmissions. For example, if 1% duty cycle is applied, a node is allowed to transmit only for 36 s per hour and stay inactive for the

* Corresponding author.
E-mail addresses: dimitrios.zormpas@tyndall.ie (D. Zorbas), khaled.abdelfadeel@ieee.org (K. Abdelfadeel), pkotzani@unipi.gr (P. Kotzanikolaou), d.pesch@cs.ucc.ie (D. Pesch).

https://doi.org/10.1016/j.comcom.2020.01.056
Received 18 October 2019; Received in revised form 10 January 2020; Accepted 25 January 2020
Available online 31 January 2020

rest of the period. The same restriction holds for the gateways. If it receives multiple data packets within a short amount of time, it will not be able to acknowledge all of them. Thus, an efficient design needs to be adopted for a reliable communication service where received packets are acknowledged without violating the regional duty cycle rules.

We believe that a time-slotted approach would be a good alternative to the LoRaWAN standard ALOHA-based MAC. This is because of the centralised nature of the network architecture of LoRaWAN. Using time-slotted communications, the collision rate can be significantly reduced, which enhances the scalability and reliability of the network as is required for most IIoT use cases. However, this comes at the expense of additional overheads to compute and disseminate the schedule in addition to ensuring global time synchronisation. As gateways run on a restricted duty cycle, disseminating the schedule represents the main issue in using a time-slotted approach in LoRaWAN. Therefore, in this work, we investigate the idea of autonomous time-slotted communications in LoRaWAN in which devices self-organise and determine their slot positions in a frame autonomously.

1.1. Autonomous slot assignment

In our previous work [8], we proposed a collision-free autonomous time-slotted communication approach for a LoRaWAN network. To achieve this, a perfect hash function (PHF) with a modulo operation is employed to convert the nodes' unique identifiers (DevEUIs) into unique slot numbers. Apart from autonomous slot assignments, where each node can calculate its own slot number with a simple modulo operation, another advantage of this approach is that only a small piece of information (i.e., the frame length) must be sent by the gateway to determine the frame size and, thus, the required number of slots. However, a major disadvantage of the approach is that the PHF is typically not minimal, that is a large number of empty slots occur when converting MAC addresses into integers. For a large number of nodes the number of empty slots gets large, which gradually diminishing the advantage of time-slotted communications. Applying a minimal PHF is not possible as this requires computing the hash function every time a new node is added into the network.

1.2. Contributions

To tackle the empty slot issue and alleviate the limitations of LoRaWAN in the context of IIoT, we propose TS-LoRa: a time-slotted approach, which supports collision-free transmissions as well as retransmissions in case of packet losses. To do this an acknowledgement grouping mechanism has been developed [9]. In TS-LoRa, every transmission of the same Spreading Factor (SF) is accommodated in a unique slot, while multi-SF parallel transmissions can take place on different frequencies to avoid collisions due to the imperfect orthogonality of the SFs [10]. TS-LoRa can run either as a stand-alone protocol (in this case a separate registration and encryption mechanism is needed) or over the existing LoRaWAN network protocol with only a few modifications on the network server side.

TS-LoRa mitigates or tackles all the aforementioned issues by sacrificing some energy due to the synchronisation and acknowledgement mechanisms. Our contributions in devising TS-LoRa are summarised as follows:

1. We present an efficient mechanism that enables nodes to autonomously and securely deduce their slot number in the frame. The proposed mechanism leaves the existing keying material already defined in LoRaWAN untouched, while it guarantees that the slot assignment will be private and collision-free.
2. We enhance TS-LoRa reliability by integrating an acknowledgement transmission mechanism integrated into the synchronisation process.
3. We describe how two IIoT requirements, high reliability and guaranteed delay, can be achieved with the proposed approach.
4. We demonstrate and assess the proposed system under different node arrangements and synchronisation schemes. All the operations are developed and evaluated using a real experimental platform.

The rest of the paper is organised as follows. Section 2 describes TS-LoRa including the registration mechanism and the time-slotted operation. Section 3 presents evaluation results while Section 4 surveys recent related works. Finally, Section 5 concludes the paper and presents ideas for future directions.

2. TS-LoRa

The LoRa based system we consider here consists of a set of nodes with sensing and communication capabilities and a gateway which collects data transmitted by nodes. We assume a traditional 1-hop star network topology where the nodes' transmissions can be received by at least one gateway.

As is the case in standard LoRaWAN, the procedure is divided into phases. In LoRaWAN two phases exist, the joining phase and the data transmission phase. An optional phase where acknowledgements may be sent by the gateway may exist. During the joining phase, the nodes first register with the gateway by exchanging a set of keys that are initially used to verify the integrity of the join request and later on to facilitate encryption of the data. The nodes wake-up periodically, take a sensor measurement, and transmit a data packet over a secure channel established using the exchanged keys. In TS-LoRa, in order to avoid collisions, transmissions are performed during specific time-slots using a time division mechanism. Moreover, TS-LoRa introduces a new last phase where the synchronisation and the transmission/reception of the acknowledgements are taking place. The phases are depicted in Fig. 1 and their functionality is described in detail in the following.

2.1. Registration & slot allocation mechanism

Following the latest specification [6], a node can join a LoRaWAN network using the Over The Air Activation (OTAA) as follows: The node sends a join request of the form: `join-request=[JoinEUI|DevEUI|DevNonce]`, where `DevEUI` is the node's unique identifier, `JoinEUI` is a random application id and `DevNonce` is a 2-octet nonce. The network server will respond with a join-accept message of the form:

$$\texttt{join}-\texttt{accept} = [\texttt{JoinNonce}|\texttt{NetID}|\texttt{DevAddr}|\texttt{DLSettings}|\texttt{RxDelay}], \tag{1}$$

where `JoinNonce` is a device specific counter (expected to never repeat itself), `NetID` is a network identifier and `DevAddr` is the end-device address.

If TS-LoRa is run over LoRaWAN, it can keep the same activation process as LoRaWAN, so no additional software or hardware is needed for its implementation. The only difference that TS-LoRa introduces takes place on the network server side and is related to the generation process of the end-device address (i.e., `DevAddr`).

In TS-LoRa, every time the gateway receives a join request, a time-slot must be associated with the joining node. The gateway keeps track of the number of reserved slots (per SF) starting from slot 0 and increasing by one for every new join request. The maximum number of slots that can be assigned is denoted by S. The associated slot could be directly communicated to the node (e.g., included in the join response), however this would require several changes on the registration design of the protocol on both server and node side (e.g., a new firmware to define new fields). In order to keep using the existing LoRaWAN protocol on the node side, TS-LoRa utilises the `DevAddr` generation process. To do so, it generates a 32 bit `DevAddr` and then it checks if Eq. (2) is satisfied. If not, a new `DevAddr` is chosen at random until

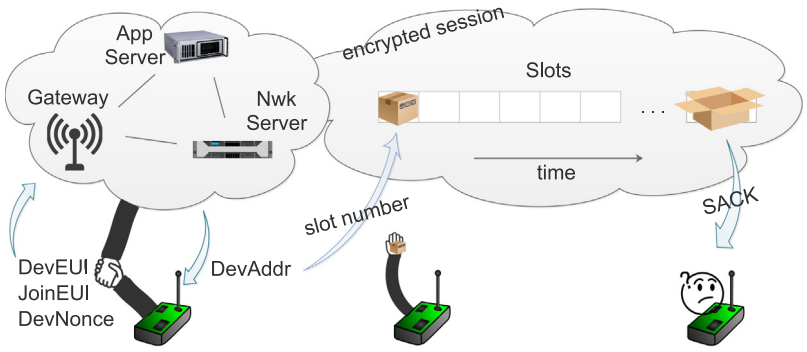


Fig. 1. Phases of TS-LoRa: network registration, data transmission, and Synchronisation/ACKnowledgements (SACK).

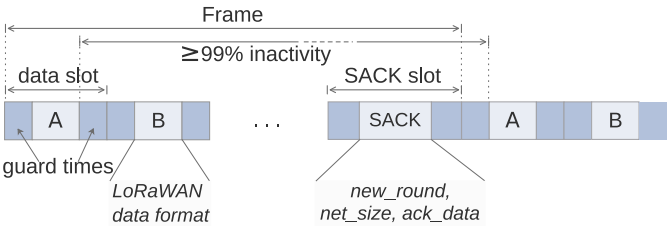


Fig. 2. Frame structure depicting two data slots (A, B) and a Synchronisation/ACKnowledgements (SACK) slot.

the desired slot number is produced. Thus, each node will be assigned to a unique slot number, which will be locally computed by each node.

$$slot = [\texttt{int}(\text{crypto\_hash}(\texttt{DevAddr}))] \ \% \ S. \quad (2)$$

It is easy to see that since `DevAddr` is randomly chosen, generating the desired slot number will only require one or few executions. We provide evidence that the slot computation requires only a few milliseconds even with a high number of nodes (*see* Section 3). In addition, this modification will not affect the underlying security services (neither positively nor negatively) since replacing a random `DevAddr` with a new one will not affect the key generation mechanisms of the protocol. In Eq. (2), crypto_hash is assumed to be a cryptographically secure hash function (e.g., SHA256). The use of a secure one-way hash function ensures that even if the slot number and S are exposed, the attacker will not be able to deduce `DevAddr`.

Once the node receives the server response, it extracts the slot number using the same equation. The process on the node side is very fast and is done only once, thus, the energy and time costs are negligible.

2.2. Data transmissions

As with all traditional time-slotted approaches, data transmissions are performed by dividing the time into repeated frames, where each frame consists of a number of slots as illustrated in Fig. 2. Each slot is capable of accommodating one transmission. Guard times are added at the beginning and at the end of each slot to tolerate small desynchronisations due to the clock drift of the nodes and propagation delays between nodes and gateway. A data slot has length equal to $T_{data} + 2g$, where T_{data} is the packet transmission time for the given SF, channel bandwidth (BW) and payload size [11], and g is the guard time.

2.3. Network synchronisation & data acknowledgements

Network synchronisation and data acknowledgements are two operations that the gateway handles using a single packet called "SACK" (Synchronisation/ACKnowledgements). We recall that synchronisation is required to ensure that data transmissions of the next frame will not violate the scheduled timings given a maximum margin defined by the guard time.

As depicted in Fig. 2, the SACK packet is transmitted in a dedicated slot at the end of each frame. All nodes must have their receiver turned on during the transmission of this packet. A SACK packet contains the start time of the next round (i.e., frame) (*new_round*), the network size in number of nodes (*net_size*), and acknowledgement data (*ack_data*). The rationale for each of these parameters is as follows:

Start time of the next round: This is an integer value measured in milliseconds to indicate the start time of the next round. Once a node receives a SACK packet, it takes some time to extract the data and perform acknowledgement related calculations. Since nodes with different hardware capabilities may be used in the network, we need to ensure that all nodes have completed these calculations before initialising a new round. Thus, this is a mostly empirical value.

Network size: The current network size is required to calculate the length of the next frame. If new nodes have been added into the network, the frame size may have to be extended to accommodate new slots. As the frame length is extended, some extra delay is added between successive transmissions. The system must be organised in this way so that it meets the application requirements of the nodes.

Acknowledgement data: In every round, the gateway keeps track of the received data packets and associates each received packet with the binary flag "1". A "0" flag is used to indicate that either nothing was received in the respective slot or that an erroneous packet was received. The gateway builds a sequence of ones and zeros whose positions in the sequence depend on the slot of each registered node. For example, if the network consists of five nodes and only the data of the node accommodated in the first slot was received, the gateway will produce an ACK binary sequence of `b'10000'`. A node that receives a SACK packet, extracts the ACK information and checks if the ith position of the sequence is a zero or a one, where i is its slot number. If it is a zero, the same packet will be repeated during the next round.

The size of a SACK packet is proportional to the number of registered nodes (of the same SF) in the network. This means that the maximum number of supported nodes in a SACK packet is about 2000 (251 Bytes).[1] Moreover, the length of the SACK slot may increase during time since more nodes of the same SF may be added to the network. In order to obey the duty cycle rules, the gateway cannot transmit another SACK packet unless a time duration of at least 99 times the duration of the previous SACK packet has passed. If T_{SACK} is the transmission time of the last SACK packet, it follows

[1] 255 Bytes is the maximum LoRa packet size (even though it is not supported by all the SFs of LoRaWAN). We assume that 4 Bytes are used to indicate the next round's start time and the network size.

that $99T_{SACK} \leq frame\_size - T_{SACK}$ must hold, where $frame\_size$ is given by the following equation (assuming 1% radio duty cycle):

$$frame\_size = \begin{cases} 100T_{data}, & \text{if } n \leq \left\lceil \frac{100T_{data}}{T_{data}+2g} \right\rceil, \\ n(T_{data}+2g)+T_{SACK}, & \text{if } n > \left\lceil \frac{100T_{data}}{T_{data}+2g} \right\rceil, \end{cases} \quad (3)$$

where n is the number of nodes, T_{data} the data transmission time, and g the guard time. In TS-LoRa the condition has been tested and it is met for any frame size.

Moreover, the SACK slot has a maximum duration which mainly depends on the transmission time of the ACK data and the guard time. If a node does not receive a SACK packet within that duration, it can still proceed to the next round assuming the same frame length as in the previous transmission. An eventual increase of the frame length that should be indicated in the non-received SACK packet, would lead to an early wake-up of the radio module and a desynchronisation during the next round. If the node is unable to synchronise with the network after two failed attempts, it postpones the next transmission until it finally receives the sync information. This also leads to a packet drop since at maximum two retries are allowed.

Unlike other synchronisation mechanisms, TS-LoRa does not require the gateway to include the global clock time into the SACK packet. The node adjusts its clock by counting the duration of the SACK slot. If it drifts over a predefined threshold compared to the normal length, it adjusts its clock accordingly. We must note that since the SACK data is sent unencrypted, some integrity and/or encryption mechanisms will be needed to avoid DDoS attacks. This is an issue that we will resolve in a future version of TS-LoRa.

2.4. Multiple parallel frames

Due to the unequal slot length per SF, TS-LoRa uses up to 6 parallel frames, one for each SF. Each frame may have its own length according to the duty cycle rules and the number of accommodating nodes. Current implementation assumes that each frame includes its own SACK slot independent of the other frames. In this case, due to the half-duplex nature of the LoRa transceiver, an additional antenna and chip is required at the gateway to transmit and schedule the SACK data if nodes with different SFs are transmitting at the same time. However, the scheduling of the SACK slots is out of the scope of the current paper.

In order to keep a high level of system reliability and determinism, we suggest to follow a multi-channel approach where each SF uses its own channel. The rationale behind this approach is to avoid inter-SF collisions due to imperfect orthogonality of SFs [10,12]. The drawback of such an approach is that the nodes must be configured to transmit only on specific channels based on the selected SF, a fact that could augment the implementation complexity. However, TS-LoRa could flawlessly work independently of the SF channel once the imperfect orthogonality is tackled by adjusting the transmit power levels [9,13].

2.5. Performance guarantees

2.5.1. Reliability

TS-LoRa can achieve high reliability mainly due to two reasons, interference avoidance and acknowledgement mechanism (re-transmissions).

Interference in LoRa is divided into intra and inter-SF interference. On one hand, intra-SF interference occurs when transmissions of the same SF overlap in time. In TS-LoRa, there is no intra-SF interference since transmissions of the same SF are accommodated in different slots. On the other hand, inter-SF interference may appear due to imperfect orthogonality of the SFs as has been theoretically [10] and experimentally [12] shown in the literature. However, in an industrial environment this issue can be solved by assigning a different frequency to each SF. LoRaWAN uses eight 125 kHz non-overlapping channels that could host all six available SFs (7–12). The only case where a packet could be lost due to interference is when packet collisions occur due to transmissions originating from an external network. However, since IIoT networks are usually deployed in a protected environment, the external distant signal would probably have much lower power compared to the internal signal. This means that due to the capture effect the stronger signal would most likely be decoded.

The acknowledgement mechanism is responsible for informing nodes about the delivery status of the last transmitted packet. On one hand, if a data packet is lost, the node can re-transmit the packet during the next round by checking the corresponding bit of the ACK sequence as described in the previous subsection. On the other hand, if a SACK packet is lost, the node considers the previously transmitted packet as lost and will transmit it again in the next round.

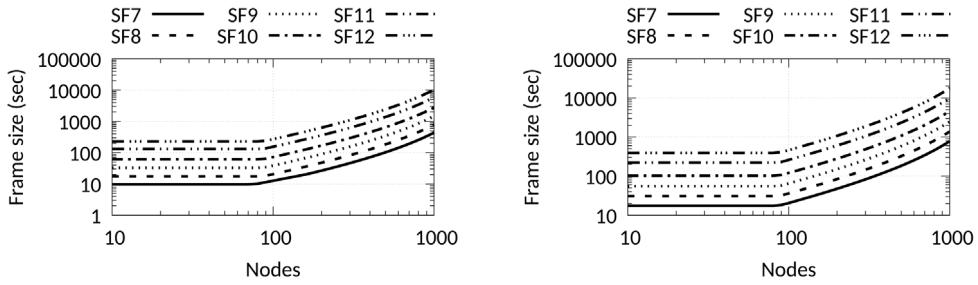
It is obvious that the data transmission mechanism in TS-LoRa is serial. This means that no new data packets are transmitted until the delivery of the most recent one has been confirmed. Since this action can block the delivery of future data, TS-LoRa sets a maximum number of re-transmissions per packet. Moreover, TS-LoRa is equipped with a reactive SF transition mechanism. Apart from that, if a node has stopped receiving SACK packages for a period of time, it has probably lost contact with the gateway. This may happen due to several reasons such as appearance of obstacles, environmental changes, mobility etc. In this case, the node can re-register with the network using a higher SF. We recall that higher SFs increase sensitivity, thus, longer ranges can be achieved. The gateway will mark the slot of the previously used SF frame as empty and will assign a different slot (with a different key) to that node. A proactive SF transmission approach as the LoRaWAN's Adaptive Data Rate (ADR) mechanism can also be applied. However, such a mechanism, and more precisely the downlink response of the gateway, should be integrated into the SACK packet, thus would require additional modifications of the LoRaWAN protocol at both network and node sides.

2.5.2. Delay

We can distinguish two types of delay in a LoRa-based network. The first type is the delay caused due to the transmission time of a data packet, which is most commonly called *time-on-air* or *airtime*. In LoRa, the airtime depends on the SF, the channel bandwidth, and the payload size. Assuming data packets of 50 Bytes and a channel bandwidth of 125 kHz, the airtime equals to approximately 10 ms and 2.3 s for SF7 and SF12, respectively [11]. These figures increase to 170 ms and 3.94 s, respectively, for a payload of 100 Bytes. It is obvious that LoRa cannot support extreme time-critical industrial applications such as the movement control of a robot since this would require less than a few milliseconds response time, which LoRa can only achieve with a very short packet, low SF, and high channel bandwidth. However, the average LoRa airtimes are fine for up to a few seconds delay tolerant applications such as massive machine shutdown, asset tracking, and smart metring systems.

The second type is the delay caused due to the radio duty cycle regulations. In fact, the duty cycle sets a lower bound on the waiting time between two successive transmissions. A typical duty cycle for the sub-GHz bands in Europe is 1% [7], which means that the nodes need to wait for the 99% of the total time. However, the regulations in Europe leave an open window as a few sub-bands do not require any channel occupation restriction (i.e., sub-bands I, J, and P).[2] For TS-LoRa, the duty cycle regulations set a minimum allowed frame size. If the number of nodes (translated into the number of slots) do not reach that frame size limit, empty slots are added to satisfy the duty cycle rules. The empty slots are filled up with transmissions, as more nodes are added to the network. Once the duty cycle frame size limit has been reached,

[2] They only impose a lower transmission power limit for these bands.



(a) 50 Bytes packet size (b) 100 Bytes packet size

Fig. 3. Frame size in seconds for different number of nodes, SFs, and packet sizes (BW = 125 kHz, guard = 20 ms).

additional nodes can be accommodated by increasing the frame size with adequate number of slots. This behaviour is illustrated in Fig. 3, where the frame size for different node populations, SFs, and packet sizes is presented.

It is obvious that the higher the number of nodes in a frame, the longer it is and, thus, the larger the delay. This practically means that if an application requires frequent packet transmission, the number of nodes must be limited to satisfy this requirement. The issue can be partially solved by moving some nodes to higher SFs but this is not always possible due to the larger transmission sparsity of higher SFs that may not always satisfy the same application requirements. Some element of admission control will be required to deal with this problem. This however, is out of scope of the current paper.

3. Evaluation & discussion of the results

3.1. Experiments setup

TS-LoRa has been implemented on a real hardware platform as a stand-alone protocol.[3] In the current work, we evaluate the key generation mechanism, the time-slotted operation as well as the synchronisation and the acknowledgement mechanism. Further implementation details can be found in [14]. We also developed an equivalent to the standard LoRaWAN ALOHA-based data transmission approach where every node transmits a packet at random times as soon as the duty cycle rules permit it. The purpose of the comparison is to show the difference in terms of packet delivery ratio compared to time-slotted LoRa.

The architecture consisted of two gateways, a Raspberry Pi 3, and 25 nodes. The first gateway was used to handle the join requests and the second for the data collection and the transmission of the acknowledgements. The Raspberry Pi played the role of the network and application server, thus it was only used to host the key generation mechanism. Both gateways have two network interfaces, the embedded IEEE802.11n to communicate with the Raspberry Pi and the LoRa interface. The gateways and the Raspberry Pi were located close to each other and their communication was done over a secure channel.

We must mention that a common issue in IIoT is the registration (and synchronisation) of a large number of devices in a short time when they boot-up for the first time. Even though data transmissions are coordinated, the registration mechanism is still ALOHA-based. Thus, powering-up all devices at the same time would cause a high number of join request collisions. A straightforward solution to this problem is to randomly wake-up the nodes such as the probability of collisions is low. However, this would lead to very long joining times. Hence, in this work, we tackle this problem by using two narrow duty-cycle-free channels (869.7 and 869.85 MHz) [7] so that a node can send multiple join requests without the – time consuming – duty cycle intervention.

[3] TS-LoRa was implemented on Pycom Lopy4 nodes using the Micropython programming language. The code is available online at https://github.com/deltazita/ts-lora/.

Table 1
Experimental parameters.

Parameter	Value
Nodes (n)	25
Bandwidth (BW)	125 kHz
Preamble symbols	8
Coding rate	4/5
Spreading factor (join request)	12
Spreading factor (data)	7–9
Frequency	EU868
Radio duty cycle	1% for data and SACKs
Join request size	20 Bytes
Data packet size	100 Bytes
Guard time	15 ms (unless specified)
Tx power (nodes)	7 dBm (5 mW) for join requests 14 dBm (25 mW) for data
Tx power (gateways)	7 dBm for request responses 14 dBm for SACK packets
Packet transmission rate	1 per 17.5 s (SF7)
Max. network size (S)	1000
Data encryption	AES-128 (ECB mode)
Hash function for Eq. (2)	SHA-256

However, this solution is not LoRaWAN-compliant and a transmission power downgrade has to be imposed (7 dBm).

The nodes were scattered in a 4-floor building as well as in the courtyard outside it. A set of tested positions along with the positions of the gateways and the Raspberry Pi are depicted in Fig. 4. The total deployment area was larger than 1100 m^2 and the furthest node was set about 35 m away from the gateways. The nodes and the gateways were equipped with a typical 2 dBi antenna. The communication between the gateways and the majority of the nodes was performed through thick concrete and stone walls, which causes significant attenuation of the signal. This actually justifies the low achieved range (max 40 m with SF7). Five of the nodes were mobile for random periods of time throughout the evaluation. These nodes were carried by people whereas the average moving time was about 45 min. We should note that co-located LoRa networks existed in the building.

The evaluation lasted 7 h and was repeated several times over different days, whereas during that 7 h period each node was able to send approximately 1500 packets (SF7). Table 1 summarises the experimental parameters.

3.2. Results

3.2.1. Joining & network synchronisation time

In the first experiment, we evaluate the nodes' joining time which is split into two phases; the request phase and the synchronisation phase.

Every time the gateway receives a join request, a DevAddr is generated and join response is then forwarded to the node. If the node does not receive a reply within a time limit, it switches to sleep mode and wakes-up later to re-transmit the request. The time needed to generate the DevAddr is important since this is the dominant factor that affects

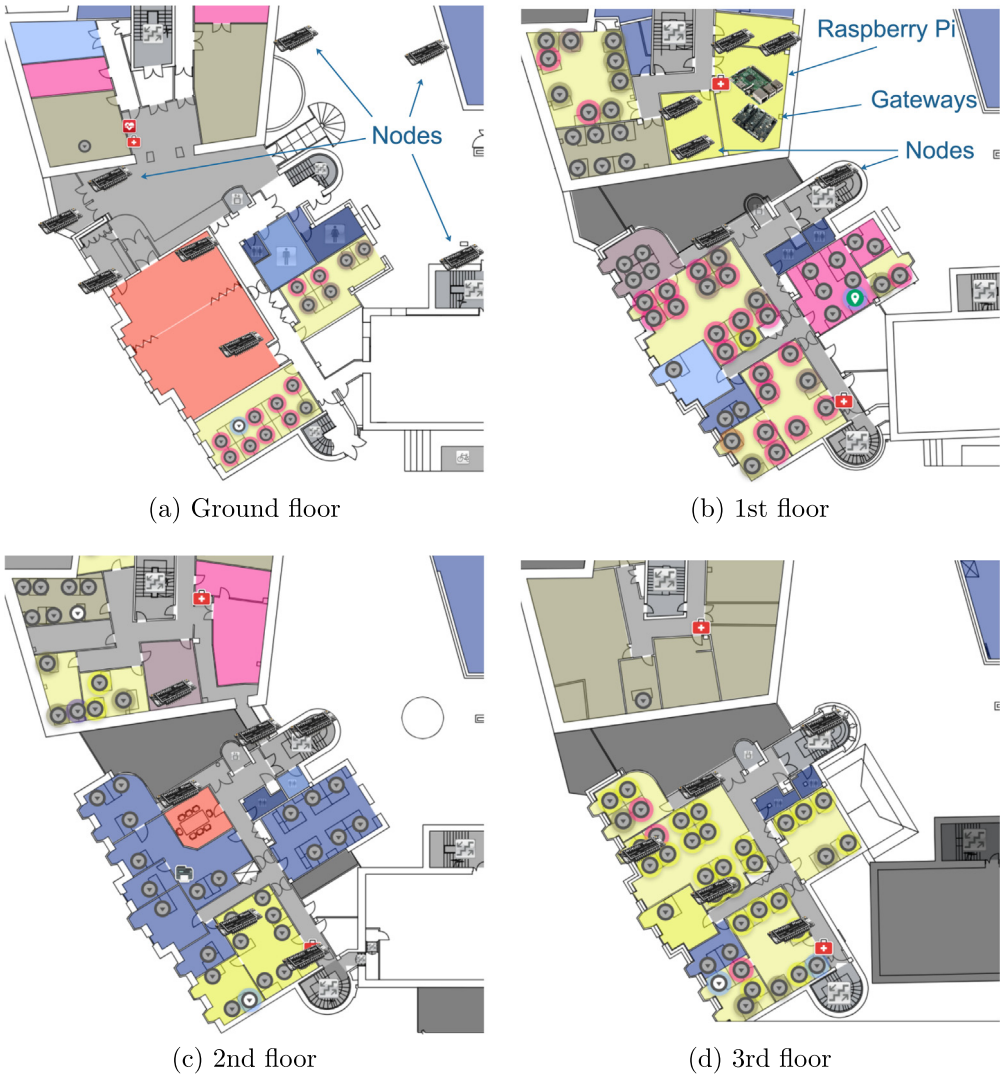


Fig. 4. Positions of the nodes and the gateways in the 4-floor building.

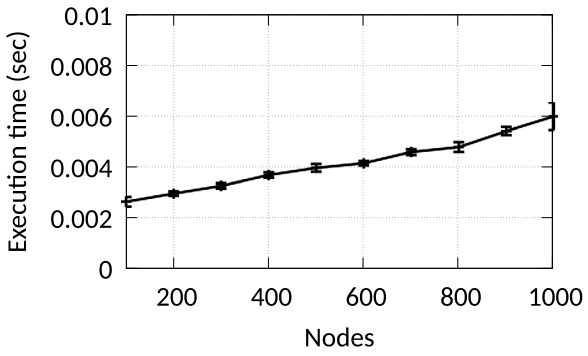


Fig. 5. Average time to generate a slot.

the node's time limit before going back to sleep mode and preserve energy. For this reason, we evaluate the average and the maximum time needed to generate a DevAddr and, thus, the desired slot number. The results are presented in Fig. 5 and they show that the average slot generation time increases linearly with the number of nodes. This is an expected behaviour due to the modulo operation of Eq. (2). However, the average execution time is 2.3 ms to 6 ms while the maximum captured value over 10 K generations was only 53 ms. Thus, the node's waiting time in this stage is negligible.

After a node joins the network, it calculates its transmission slot (based on the information it receives from the gateway) and waits for a SACK packet. The slot calculation is negligible compared to the average SACK waiting time. Thus, the duration of this phase depends on the length of the frame which – as we explained previously in the text – is mainly affected by the duty cycle rules or the number of nodes for the specific SF. This means that, assuming that the nodes request to join the network at random times, the average synchronisation time is half the length of the frame. The average synchronisation time in the experiments was approximately 9.5 s (SF7) which is very close to half of the round length presented in Fig. 3.

3.2.2. Clock synchronisation

In this subsection, we evaluate the synchronisation mechanism and we describe how to choose a safe guard value for a specific node population. A safe guard time is the shortest guard time that does not allow overlaps between neighbouring slot transmissions. Thus, the guard time must be – in the worst case scenario – equal to the maximum clock drift of a node during a single round. The maximum clock drift of a typical crystal oscillator is about 100ppm, which results in a clock drift of 0.1 ms per second (~ 1.75 ms in total for a SF7 frame). We also experimentally found that a node needs approximately 7 ms to switch radio mode while another 3 ms is needed for processing the SACK

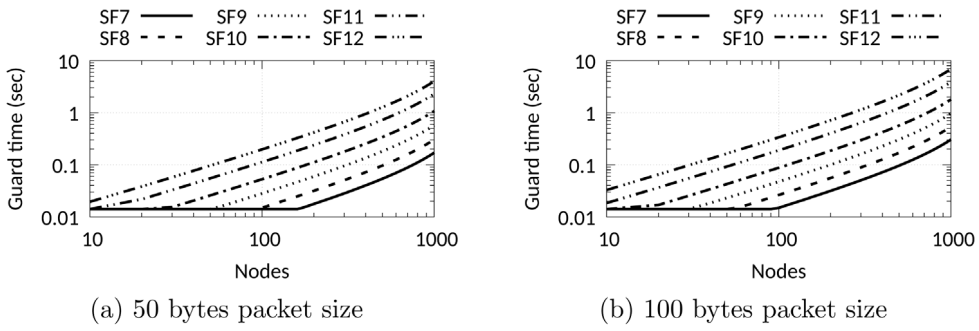


Fig. 6. Theoretical guard time for different node populations based on the experimental data.

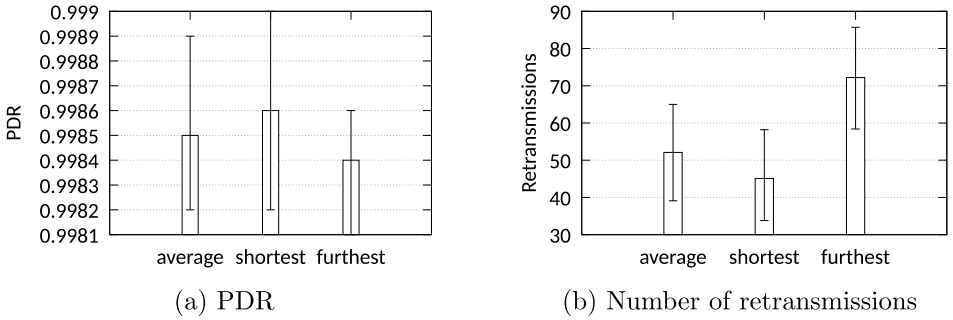


Fig. 7. Packet delivery ratio and number of retransmissions for the entire network as well as for the best and worst case scenarios.

packet. Thus, a total time of 11.75 ms needs to be taken into account for a SF7 frame. However, taking into account that the maximum number of re-transmissions per original transmission is 2, we must ensure a maximum desynchronisation of 3 successive frames, hence, a guard time of approximately 15 ms (i.e., $3 \times 1.75+7+3$). However, the guard time should be adjusted accordingly for higher SFs or for node populations over the duty cycle limit due to the longer length of the frames. Assuming a linear maximum clock drift over time, we can compute the required guard time for different frame lengths and SFs. Those guard times are shown in Fig. 6. We can observe that in case of hundreds of nodes, the guard times become substantial, however, we expect that only a few nodes will be accommodated in higher SFs due the high application duty cycle of those SFs.

3.2.3. Packet delivery ratio

In this subsection we report on the overall achieved packet delivery ratio (PDR), the number of re-transmissions, and the dropped packets throughout the experiments. We measured the overall average values in the network as well as the values of two particular nodes; one with the best and one with the worst RSSI values. The first node was placed 2 m away from the gateway with LOS and an average RSSI value of −15 dBm, while the second one was placed about 35 m away, non-LOS, with an average RSSI of −102 dBm which is close to the sensitivity limit (SF7 settings). We measured the PDR over SF7-9, however, the results differed only slightly from each other (a slightly higher number of retransmissions on the long links of SF7 due to the lower sensitivity).

The results are depicted in Fig. 7 and they reveal a very strong and robust performance of TS-LoRa for all tested SFs. We can observe that even the node with the worst link achieved more than 99.8% PDR while the number of retransmissions were kept very low for all nodes. Moreover, the corresponding non-confirmable ALOHA-based approach did not achieve a PDR higher than 68% due to the high number of collisions.

3.2.4. Energy consumption

In order to measure the energy consumption of the system, we used a power analyser providing a constant voltage of 3.5 V. We measured the power consumption of the entire device[4] The LoRa module of the nodes was turned on only for the join request, the synchronisations, and the data transmissions.

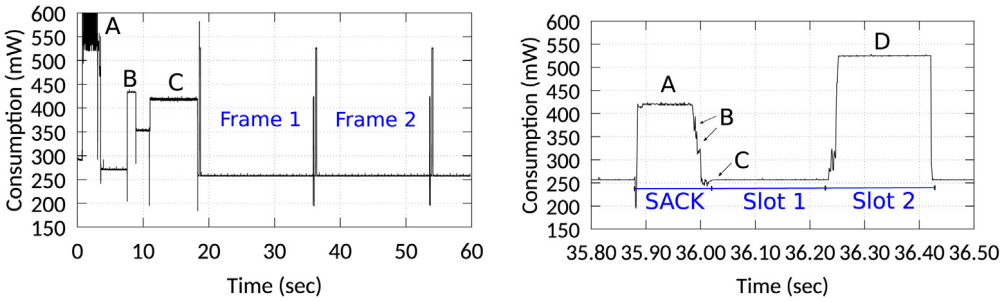
Fig. 8a presents the energy consumption of a node for the first 60 s once it was plugged into power. We can clearly see the different stages of the procedure such as the registration mechanism followed by consecutive frames. The three short energy peaks in the figure correspond to data transmissions at the beginning of each frame. The power consumption during the join request transmission is less than that of a data transmission since the transmission power is lower.

Fig. 8b is a zoom-in figure shedding some light on the SACK reception and data transmission times that occur in the time interval between the end of the first frame and the beginning of the second frame. We can distinguish the consumed power for the SACK reception, its processing, and the transmission of the next data packet in slot 2. The energy consumption during the SACK slot is up to approximately 2 times lower than that of a data slot (45.15 mJ against 92.75 mJ). This means that the synchronisation and acknowledgements mechanism adds about 50% more energy cost compared to the typical non-confirmable LoRaWAN. However, TS-LoRa is expected to be much more efficient than the confirmable version of LoRaWAN since the latter (a) uses two downlink slots for acknowledgements and (b) usually requires multiple re-transmissions due to the radio duty cycle restriction at the gateway.

3.3. Simulation results

We also conduct a set of simulations to assess the behaviour of TS-LoRa in scenarios with many nodes. We compare our approach with the standard LoRaWAN to support a confirmable application. In this case, each uplink transmission has to be acknowledged by the gateway to confirm the reception. Otherwise, a re-transmissions is scheduled up to 8 times. For acknowledgements, one of the two receive windows, Rx1

[4] Some board components such as the LED, the battery charging module, and the RTS/CTS controller, were turned on during the measurements. However, those components can be turned off in a commercial deployment to save energy.



(a) First minute: (A) node initialisation, (B) join request, (C) synchronisation waiting time

(b) SACK reception and data transmission: (A) SACK reception, (B) SACK processing, (C) end of 1st frame, (D) data transmission of 2nd frame

Fig. 8. Energy consumption of a node accommodated in the 2nd slot throughout different stages of the experiment.

Table 2
Simulation parameters.

Parameters	Value
Random seeds	10
Devices	10–1000 randomly scattered
Cell range	500 m (city deployment)
Bandwidth	125 kHz
Coding rate	4/5
Spreading factor	7–12
Concurrent receptions	8
Channels	8 Uplink/Downlink with 1% duty cycle plus 1 Downlink with 10% duty cycle
Data periodicity	Based on Eq. (2)
Retransmissions	8 times before dropping
LoRaWAN MAC header	7 Bytes
LoRaWAN ACK	0 Bytes
TS-LoRa ACK	$4 + \lceil \frac{1\text{bit per occupied slot}}{8} \rceil$ Bytes
Guard time (TS-LoRa)	Based on Eq. (2)
Path loss model [15]	$\overline{L_{pl}}(d_0) = -127.41$ dBm $d_0 = 40$ m, $\gamma = 2.08$, $\sigma = 5$
Payload	100 Bytes
Voltage	3.5 V
Power consumption (transmission)	76 mA (experimental)
Power consumption (reception)	46 mA (experimental)
Receiver sensitivities (per SF)	[−123,−126,-129,−132,−134.53,-137] dBm
Capture effect threshold	6 dBm
Packet error model	See [9]

or Rx2, after each uplink transmission, is used. The receive windows start with a delay of 1 and 2 s respectively after the data transmission as it is defined in the standard [6]. We assume that the nodes are scattered around the gateway and their SFs are computed based on their Euclidean distance to the gateway. All other simulation parameters are summarised in Table 2. Both approaches are developed using the LoRaFREE simulator [9]. The simulator is developed in Python and takes into account a packet error model, the imperfect orthogonality of spreading factors, the signal fading effect, the capture effect, and the duty cycle limitation at both the nodes and the gateway. For fair comparison purposes, we assume that the data generation periodicity of LoRaWAN follows the frame size length of TS-LoRa. This means that less data is generated per time unit as we increase the SF value and the number of nodes as it is derived from Eq. (3). We must note that this is in favour of LoRaWAN as a fixed data generation periodicity would lead to systematic collisions, resulting in worse performance.

Fig. 9 presents the packet delivery ratio and the total energy consumption of TS-LoRa and LoRaWAN (confirmable application) for variable node numbers. The results show that TS-LoRa achieves an up to 99% improvement in terms of PDR while keeping the energy consumption levels lower than that of LoRaWAN. The delivery ratio of LoRaWAN slightly increases after 500 nodes. This is because the data generation periodicity with higher number of nodes is sparser, thus, after a certain point the number of collisions decreases. The difference in energy consumption is mainly due to the high number of non-acknowledged packets in LoRaWAN which causes significant overhead due to re-transmissions. The reason behind this is the limited duty cycle of the gateway. As depicted in Fig. 10 the number of non-acknowledged packets increases linearly with the number of nodes. Finally, the number of lost packets presents the same trend as shown in Fig. 10b. Apparently, this is due to the Aloha-based nature of LoRaWAN, leading to a massive number of collisions. On the contrary, in TS-LoRa, we observed that all the lost packets were due to the path-loss effect of the links (i.e. channel fading).

4. Related work

Time-slotted communications is a well-known MAC approach to alleviate collisions, which enhances network scalability and reliability. However, it is often regarded as impractical because of the costs associated with computing and disseminating the transmission schedule as well as the required global synchronisation. These concerns are even more valid in LoRaWAN because of the targeted low-power applications and the limited restrictions on radio duty-cycle and downlink availability. In addition to that, LoRaWAN usually targets sensor devices that have limited computation capabilities. As these network limitations have not experienced before in any wireless sensor network, the time-slotted solutions in the literature cannot directly be adopted in LoRaWAN. For this reason, novel techniques to effectively enable time-slotted transmission over LoRa, especially schedule computation and synchronisation, are required.

A number of solutions have recently been proposed in the literature to consider a time-slotted approach [9,16–21]. A centralised scheduling algorithm to compute spreading factors, channels, and time slots as well as improve LoRaWAN scalability was proposed by Lee et al. [16]. The schedule is leveraged only by class B nodes as synchronisation relies on class B beacons. The overhead of schedule dissemination is neglected, which questions the applicability of the study. Reynders et al. [17], propose to organise transmissions into frames and sub-frames. The gateway transmits a beacon before every frame to synchronise the transmissions and guide the spreading factor allocation process. If a node wants to send a packet, it has to wait for a beacon first. Despite the time-frame structure, the access within sub-frames is still ALOHA-based, thus, collisions are not eliminated. In [18], the synchronisation process is initiated by the nodes and the gateway centrally calculates the complete schedule for each node. Here, the schedule is calculated based on the application requirements, such as the data periodicity, and it is sent back to nodes using a specific probabilistic data structure, called Bloom filter. Due to the probabilistic nature of these structures, multiple nodes share the same slot with a certain probability and,

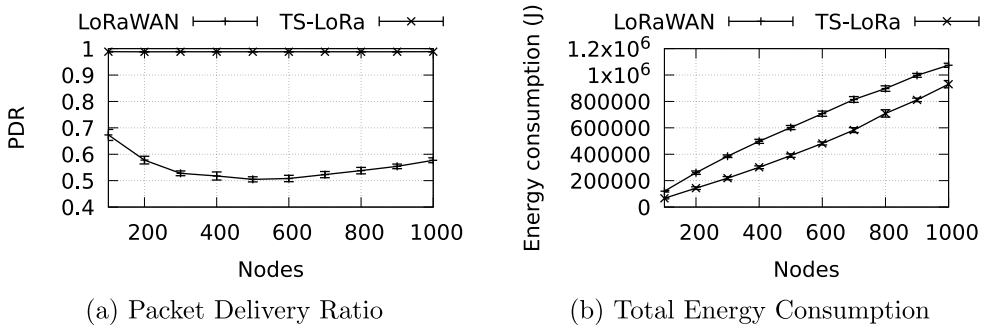


Fig. 9. Packet delivery ratio and energy consumption for variable number of nodes.

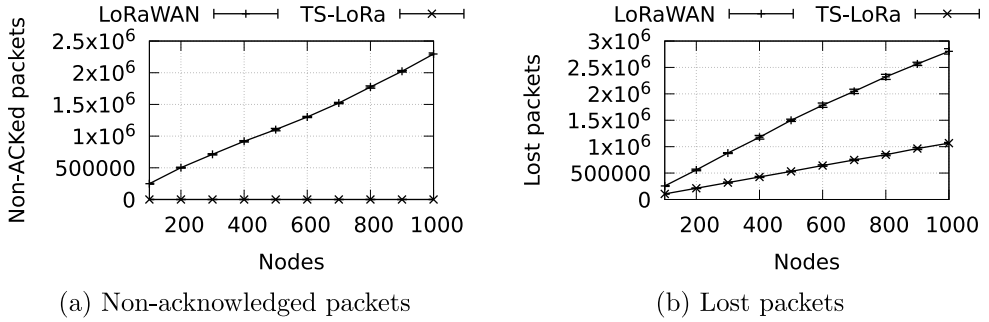


Fig. 10. Number of non-acknowledged and lost packets for variable number of nodes.

therefore, this technique does not eliminate collisions either. Moreover, these approaches have not been experimentally verified.

In [19], low-cost localisation devices (GPS for outdoor and UWB for indoor) are used together with LoRaWAN nodes to opportunistically obtain time synchronisation, which is used to implement an efficient scheduled medium access strategy. This technique improves the throughput compared to standard LoRaWAN but requires additional hardware and exhibits higher energy consumption. Ebi et al. [20] consider time-slotted communications to enable repeater nodes in LoRaWAN to extend the coverage in harsh environments such as in underground communications. A predefined periodic cycle is shared among all nodes, which consists of predefined times for synchronisation, uplink, and downlink. The synchronisation is performed via periodic beacons by the repeaters, where repeaters are assumed to have access to at least one external time source to synchronise their internal real-time clock to the coordinated universal time. In [21], a time-slotted multi-channel MAC protocol is proposed, where a predefined frame structure is shared among all nodes. In order to achieve that a synchronisation algorithm is proposed that uses the difference between timestamps of the synchronisation packets to capture the time synchronisation errors on all channels. The synchronisation is shown to be under *5usec*, however, a study looking at the overhead was missing. Albdelfadeel et al. [9] consider a time-slotted approach to enable bulk data collection in LoRaWAN. In this work, an overhead phase is required for joining and synchronisation before each data collection. As a result, significant improvements in terms of data delivery ratio and device lifetime are achieved. The authors also propose a centralised approach for the same problem [22]. The last two works are collision-free but they do not support real-time data collection.

Most of the aforementioned studies implement centralised coordination of time-slotted communications, where the gateway assigns separate slot(s) to all nodes. This approach requires an overhead phase for the gateway to disseminate the computed schedule. In contrast, in this work, we have proposed a mechanism for the nodes to deduce their slots in the frame autonomously with minimal information assistance by the gateway. This is done using the node address that is sent by the gateway during the joining phase. For time synchronisation, a dedicated slot is used in each frame. Also, the gateway uses the same slot to acknowledge received packets in the frame to overcome the duty cycle restriction.

5. Conclusion & future work

5.1. Conclusion

In this work, we proposed TS-LoRa as a novel time-slotted communications approach over LoRaWAN. TS-LoRa enables nodes to self-organise the time slot schedule within frames. For this purpose, the network server and the nodes share an easy-to-compute hash algorithm. This algorithm maps the nodes' addresses that are assigned during the join phase into unique slot numbers. The mechanism ensures backward compatibility with legacy LoRaWAN nodes and liberates TS-LoRa from the huge overhead of the schedule dissemination. In TS-LoRa, the only information that the network server has to send is the frame length. As this is the same piece of information for all nodes, the network server broadcasts it to all nodes at the same time. This makes TS-LoRa scalable in contrast to other proposed time-slotted approaches for LoRaWAN in the literature. Additionally, TS-LoRa dedicates the last slot in each frame for sending the "SACK" packet, which handles time-synchronisation and acknowledgements. For the acknowledgements, the network server groups multiple acknowledgements in a single packet to confirm receptions from all slots at once. We experimentally evaluated TS-LoRa and the results revealed a very high packet delivery ratio for all the tested spreading factors. As for all the confirmable applications an extra energy cost to maintain acknowledgements exists. However, as the simulation results revealed, this cost is lower than LoRaWAN when supports a confirmable traffic.

5.2. Open issues & future work

The extra energy cost of synchronisation is the biggest issue when designing synchronous communication protocols. In our experiments the SACK mechanism caused a roughly 50% increase in total energy consumption. Although one might argue that this extra cost is justified

by a much higher PDR, it can be scaled down multiple times by considering a sparser SACK transmission. In this case, multiple ACKs for previous transmissions can be combined in a single SACK packet. The data can be encoded appropriately so that a minimum amount of bytes is used (e.g., using run-length encoding). We should mention, however, that this approach would increase delay in case of re-transmissions.

A key weakness of LoRa is the long transmission time for high SFs, causing very long frame lengths in TS-LoRa and, thus, a long delay between successive transmissions. Depending on the application, a number of nodes would never use all the available frames to transmit data and could eventually leave other nodes to use those slots. This implies that nodes share the same slots, which may cause collisions. Nevertheless, given the nodes past activity the system could adapt the shared slot assignment mechanism to achieve a very low probability of collisions.

Finally, another issue caused by the delay due to long frames is that the system may not fulfil the requirements for supporting urgent traffic (e.g., a few millisecond long packet for alarm purposes). An interesting solution to that problem would be to send such packets using a separate dedicated ALOHA channel. A burst of packets can be sent using optimal SF settings [12] to increase the probability of delivery.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Dimitrios Zorbas: Conceptualization, Methodology, Validation, Writing - original draft, Writing - review & editing, Project administration, Software, Resources, Investigation. **Khaled Abdelfadeel:** Methodology, Writing - original draft, Writing - review & editing. **Panayiotis Kotzanikolaou:** Methodology,Writing - original draft, Writing - review & editing. **Dirk Pesch:** Writing - original draft, Writing - review & editing, Supervision.

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, the CONFIRM fund under Grant Number 16/RC/3918, and the European Union's Horizon 2020 research and innovation programme under the Marie Skodowska-Curie grant agreement No. 713567.

References

[1] B. Martinez, C. Cano, X. Vilajosana, A square peg in a round hole: The complex path for wireless in the manufacturing industry, IEEE Commun. Mag. 57 (4) (2019) 109–115.

[2] FieldComm Group, WirelessHART Specification 75: TDMA data-link layer, 2020, https://fieldcommgroup.org/hart-specifications, Online; accessed 7-January-2020.

[3] IEEE time-sensitive networking task group, 2019, http://www.ieee802.org/1/pages/tsn.html, Online; accessed 13-October-2019.

[4] T. Watteyne, M. Palattella, L. Grieco, Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement, in: RFC, vol. 7554, IETF, 2015.

[5] Semtech Corporation, AN1200.22, LoRa™ modulation basics, 2015, www.semtech.com/uploads/documents/an1200.22.pdf, Online; accessed 10-October-2019.

[6] LoRa Alliance Technical Committee, LoRaWAN™ 1.0.3 specification, 2018, , Online; accessed 17-October-2019.

[7] ETSI EN 300 220-2, Short range devices (SRD) operating in the frequency range 25 MHz to 1 000 mhz; part 2: Harmonised standard for access to radio spectrum for non specific radio equipment, 2018, p. V3.2.1, www.etsi.org/deliver/etsi_en/300200_300299/30022002/03.02.01_30/en_30022002v030201v.pdf, Online; accessed 18-October-2019.

[8] D. Zorbas, B. O'Flynn, Autonomous collision-free scheduling for LoRa-based industrial internet of things, in: 20th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), IEEE, 2019, pp. 1–5.

[9] K.Q. Abdelfadeel, D. Zorbas, V. Cionca, D. Pesch, FREE–Fine-grained scheduling for reliable and energy efficient data collection in LoRaWAN, IEEE Internet Things J. 7 (1) (2020) 669–683.

[10] D. Croce, M. Gucciardo, S. Mangione, G. Santaromita, I. Tinnirello, Impact of LoRa imperfect orthogonality: Analysis of link-level performance, IEEE Commun. Lett. 22 (4) (2018) 796–799.

[11] Semtech Corporation, LoRa modem design guide, 2013, www.semtech.com/uploads/documents/LoraDesignGuide_STD.pdf, Online; accessed 17-September-2019.

[12] D. Zorbas, P. Maillé, B. O'Flynn, C. Douligeris, Fast and reliable LoRa-based data transmissions, in: IEEE Symposium on Computers and Communications (ISCC), IEEE, 2019, pp. 1–6.

[13] K.Q. Abdelfadeel, V. Cionca, D. Pesch, Fair adaptive data rate allocation and power control in LoRaWAN, in: 2018 IEEE 19th International Symposium on "a World of Wireless, Mobile and Multimedia Networks" (WoWMoM), 2018, pp. 14–15.

[14] D. Zorbas, Design considerations for time-slotted LoRa(WAN), in: International Conference on Embedded Wireless Systems and Networks (EWSN), 1st Workshop on Massive LoRa Deployments: Challenges and Solutions, ACM, 2020, pp. 1–6.

[15] M.C. Bor, U. Roedig, T. Voigt, 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, in: MSWiM '16, ACM, 2016, pp. 59–67.

[16] J. Lee, W. Jeong, B. Choi, A scheduling algorithm for improving scalability of LoRaWAN, in: International Conference on Information and Communication Technology Convergence (ICTC), IEEE, 2018, pp. 1383–1388.

[17] B. Reynders, Q. Wang, P. Tuset-Peiro, X. Vilajosana, S. Pollin, Improving reliability and scalability of LoRaWANs through lightweight scheduling, IEEE Internet Things J. 5 (3) (2018) 1830–1842.

[18] J. Haxhibeqiri, I. Moerman, J. Hoebeke, Low overhead scheduling of LoRa transmissions for improved scalability, IEEE Internet Things J. 6 (2) (2018) 3097–3109.

[19] F. Bonafini, A. Depari, P. Ferrari, A. Flammini, M. Pasetti, S. Rinaldi, E. Sisinni, M. Gidlund, Exploiting localization systems for LoRaWAN transmission scheduling in industrial applications, in: 15th IEEE International Workshop on Factory Communication Systems (WFCS), IEEE, 2019, pp. 1–8.

[20] C. Ebi, F. Schaltegger, A. Rüst, F. Blumensaat, Synchronous LoRa mesh network to monitor processes in underground infrastructure, IEEE Access 7 (2019) 57663–57677.

[21] S. Gao, X. Zhang, C. Du, Q. Ji, A multichannel low-power wide-area network with high-accuracy synchronization ability for machine vibration monitoring, IEEE Internet Things J. 6 (3) (2019) 5040–5047.

[22] D. Zorbas, K.Q. Abdelfadeel, V. Cionca, D. Pesch, B. O'Flynn, Offline scheduling algorithms for time-slotted LoRa-based bulk data transmission, in: IEEE 5th World Forum on Internet of Things (WFIoT), IEEE, 2019, pp. 1–6.