Low Overhead Scheduling of LoRa Transmissions for Improved Scalability

Jetmir Haxhibeqiri, Ingrid Moerman, and Jeroen Hoebeke

***Abstract*—Recently, LoRaWAN has attracted much attention for the realization of many Internet of Things applications because it offers low-power, long-distance, and low-cost wireless communication. Recent works have shown that the LoRaWAN specification for class A devices comes with scalability limitations due to the ALOHA-like nature of the MAC layer. In this paper, we propose a synchronization and scheduling mechanism for LoRaWAN networks consisting of class A devices. The mechanism runs on top of the LoRaWAN MAC layer. A central network synchronization and scheduling entity will schedule uplink and downlink transmissions. In order to reduce the synchronization packet length, all time slots that are being assigned to an end node are encoded in a probabilistic space-efficient data structure. An end node will check if a time slot is part of the received data structure in order to determine when to transmit. Time slots are assigned based on the traffic needs of the end nodes. We show that in case of a nonsaturated multichannel LoRaWAN network with synchronization being done in a separate channel, the packet delivery ratio (PDR) is easily 7% (for SF7) to 30% (for SF12) higher than in an unsynchronized LoRaWAN network. For saturated networks, the differences in PDR become more profound as nodes are only scheduled as long as they can be accommodated given the remaining capacity of the network. The synchronization process will use less than 3-mAh extra battery capacity per end node during a one year period, for synchronization periods longer than three days. This is less than the battery capacity used to transmit packets that are going to be lost in an unsynchronized network due to collisions.**

***Index Terms*—Bloom filters, LoRa, LoRaWAN, probabilistic data structures, scheduling, synchronization.**

I. Introduction

THE INTERNET of Things (IoT) is finding its way into different domains ranging from environmental monitoring, building automation, logistics, smart cities, etc. This diversity of domains where IoT is being applied also brings diversity in terms of the applications that have to be supported. The forecast of the number of end devices that will be connected by the end of the decade says that there will be up to 20.8 billion end nodes, compared to 6 billion currently deployed [1]. A majority of these devices will use wireless technology to connect to the backbone network, requiring highly scalable wireless networks in order to serve such high numbers of end nodes. Moreover, many of these devices will be battery powered and will only require low data rate and low power communication. The type of networks supporting such use cases are called low power wide area networks (LPWANs). Today, different LPWAN technologies exists, such as SigFox [2], NBIot [3], LoRaWAN [4], weightless [5], etc.

One of the LPWAN technologies that has gained great interest in recent years is LoRaWAN. LoRaWAN builds on top of the LoRa physical layer, which has been patented by Semtech [6]. The combination of LoRaWAN at the MAC layer and LoRa at the physical layer makes it possible for end devices to only consume little power and communicate over long distances up to several kilometers. A number of LoRaWAN networks have already been deployed in different countries, but there are still doubts in the research community regarding the scalability of these networks [7].

Different aspects of LoRaWAN are already being studied by the research community such as network scalability, self-interference, spreading factor (SF) orthogonality, etc. Until now, to the best of our knowledge, only very few studies are addressing how to improve the weak points of LoRaWAN. We believe that by properly scheduling transmissions of end nodes, we can boost up network scalability and traffic reliability significantly, without changing the MAC behavior of the end nodes. When done right, the power that is currently used to transmit packets that will never arrive at the gateway due to collisions can be used to perform the required synchronization and scheduling, not increasing the overall energy consumption.

In this paper, we present the design of a low overhead fine-grained synchronization and scheduling scheme for LoRaWAN networks, where the timing and amount of transmissions of end devices is dictated by a central entity that resides in the network, preferably at the network server. This entity schedules transmissions of end devices by sending a list of time slot indexes when they are allowed to transmit. These indexes are encoded in a probabilistic data structure using Bloom filters. This reduces the size of the messages that are needed to perform the synchronization and scheduling.

The remainder of this paper is organized as follows. In Section II, we will give an introduction to LoRaWAN networks and their limitations with respect to network scalability, followed by an overview of related work in Section III. Section IV motivates our approach, followed by a detailed description of our novel scheduling scheme in Section V.

Manuscript received April 11, 2018; revised June 19, 2018, September 6, 2018, and October 12, 2018; accepted October 28, 2018. Date of publication October 31, 2018; date of current version May 8, 2019. This work was supported by the Flemish FWO SBO through the Intelligent Dense and Long Range IoT Networks Project under Grant S004017N. *(Corresponding author: Jetmir Haxhibeqiri.)*

The authors are with the Internet and Data Laboratory Research Group, Department of Information Technology, Ghent University—imec, 9052 Ghent, Belgium (e-mail: jetmir.haxhibeqiri@ugent.be; ingrid.moerman@ugent.be; jeroen.hoebeke@ugent.be).

Digital Object Identifier 10.1109/JIOT.2018.2878942

Section VII presents the analysis of the resulting performance together with an evaluation of the battery usage overhead of the proposed scheme. Finally, Section VIII concludes this paper and discusses some possibilities on how this paper can be extended.

II. LoRaWAN and Its Limitations

The LoRaWAN technology can be separated into two parts: 1) the LoRa physical layer, that has been patented by Semtech [6] and 2) the MAC layer protocol and network system architecture, called LoRaWAN, designed by the LoRa Alliance [8].

For the LoRa physical layer, spread spectrum modulation and forward error correction techniques are used to make the communication robust against noise and interference and to increase the receiver sensitivity. Each bit of information is represented by multiple chips of information that are transmitted over a 125-kHz LoRa channel. By increasing the SF, the number of chips per symbol is increased, thereby decreasing the nominal data rate. Six different SFs are used, ranging from 7 to 12, that are orthogonal to each other [6]. The selection of an SF is a tradeoff between coverage range and data rate. The higher the SF, the higher the coverage and lower the data rate is. The number of chips per symbol is calculated as 2^{SF}. Forward error correction codes with code rates 4/5 up to 4/8 are used to find erroneous bits, while diagonal interleaving is used to make the communication robust against burst interference.

LoRa networks can operate in the 433-, 868-, or 915-MHz frequency bands. In Europe, only the 868- and 433-MHz bands can be used. In the 868-MHz band, there are three 125-kHz channels that are mandatory to be supported by every end device. There are another five 125-kHz channels in the 867-MHz sub-band that can be used for LoRa communication [4], with 1% duty cycle per sub-band and 14-dBm transmit power. Optionally, if the high power high duty cycle 125-kHz channel at 869.52 MHz is used (10% duty cycle, 20-dBm tx power), then only four channels from the 867-MHz sub-band can be used [9].

In addition to the robust LoRa physical layer, the LoRaWAN MAC layer provides the medium access control mechanism that enables communication between multiple end devices and their gateway(s). A star topology is used for LoRaWAN networks, consisting of one or more gateways that relay traffic between end devices and a central network server. The network server manages the end devices and the gateways and is responsible for de-duplicating the traffic in uplink and scheduling downlink transmissions, if needed.

The LoRaWAN standard defines three classes of end devices, namely A, B, and C. In class A devices, any uplink transmission is followed by two receiving downlink windows that are opened 1 and 2 s, respectively, after the end of the uplink transmission. It is the responsibility of the network server to schedule the downlink traffic at the exact time and to perform the timing control. Transmission in the second receive window happens in the high power high duty cycle channel (869.52 MHz) using SF12 to maximize the chances for reception [10]. There is no possibility of downlink communication without uplink triggering, so every downlink communication has to wait for a preceding uplink communication. In Europe, if communication happens in the 868-MHz band, both gateways and end devices have to comply with a duty cycle limit of 1%.

One of the main concerns for LoRaWAN networks is their scalability. As the LoRaWAN MAC layer for class A devices behaves as an ALOHA-like MAC protocol, there is no mechanism to ensure the reliability of communication and to boost up the scalability of the network. Currently, different methods are used to increase the reliability such as asking for a confirmation of reception by the network, sending multiple times the same packet in different channels, using adaptive data rate (ADR), etc. The first approach is not convenient for single gateway networks due to duty cycle limitations in downlink, which prevent a large number of end devices to be served in downlink. In the second case, multiple transmissions of the same packet in different channels results in a decrease in available capacity. Moreover, when high SFs are used, the interference will increase due to the long time on air of the packets.

Class A end devices consume the lowest power as they are asleep most of the time and are the basic set of features that each LoRaWAN end device needs to implement. In this paper we only consider class A devices. For other classes of end devices, we refer the reader to the LoRaWAN standard document in [8].

III. Related Work

LoRa technology and LoRaWAN networks have received attention by the research community in recent years as one of the enabling wireless technologies of IoT. Different studies have been published on LoRaWAN network scalability and reliability. A number of studies have investigated the capacity and scalability by modeling LoRaWAN networks as pure ALOHA networks [11]–[13], while others [7], [14]–[16] did not use the ALOHA model for LoRaWAN analyses. However, analyses based on ALOHA-like models underestimate the capacity of LoRaWAN networks by failing to adequately asses the impact of interference in the network.

In [14], a mathematical model to evaluate the packet error rate based on offered load is presented. A simulation model that is based on real interference measurements is presented in [15]. Further, in [16] a scalability analysis of LoRaWAN networks is performed using a LoRa error model together with the LoRaWAN MAC protocol in the NS-3 network simulator. In [7], a scalability study for LoRaWAN based on a stochastic geometry framework is presented. They show that the coverage probability drops exponentially with an increasing number of end devices. Two methods for decreasing the internetwork interference and for improving the reception rate are the usage of directional antennas and the usage of multiple base stations. The impact of these two methods in decreasing the internetwork interference in LoRaWANs is studied in [17].

So far, a limited number of techniques to improve scalability issues have been proposed, mostly by improving the SF

assignment to nodes [18]–[21]. In [18], the SFs and power transmissions are assigned to nodes by minimizing the collision rate within the same SF. This results in a higher PDR for end nodes at the periphery of the network. In [19], SFs are assigned by equalizing the time on air of packets sent by each node. In [20], the ADR algorithm is improved by considering the average SNR values of the last 20 transmissions for determining the SF for the end node. This increases the PDR of the network for about +50%.

Other studies show the benefits of synchronized transmissions, however, without implementing any algorithm. Rizzi *et al.* [22] showed how a synchronization scheme is able to boost up LoRaWAN network scalability by introducing a TSCH-like scheduling system for LoRa. Mikhaylov *et al.* [11] showed the theoretical LoRaWAN capacity as a function of the number of end nodes per gateway under perfect synchronization assumptions. So, to the best of our knowledge, no existing work has designed and evaluated a low overhead synchronization and scheduling solution for LoRaWAN class A devices that can further improve scalability in LoRaWAN networks, which is the key contribution of this paper and which will be discussed in the following sections.

IV. Motivation

The LoRaWAN MAC layer for class A devices uses pure ALOHA as channel access technique. This channel access technique is power efficient, as it does not require any "listen before talk" mechanism. As a downside, the number of collisions in the network will increase with an increasing number of nodes. This will decrease the total throughput of the network and will increase the average power usage per delivered packet of end devices.

One way to improve the network scalability, one might think, is using slotted ALOHA, where time is globally synchronized and transmissions may only take place in slots. However, this approach is far from optimal for the following reason. In [15], we showed that different packet parts have different degrees of importance when it comes to interference. For example, in case the preamble and header of a LoRaWAN packet are interfered then the packet cannot be decoded correctly. By applying slotted ALOHA, we will let all interfering transmissions start at the same time ($\pm$ the clock drift accuracy) while they will have different end times for different packet lengths. As such, the parts that will be most affected by interference, are at the beginning of the packet, i.e., preamble and header, meaning that the whole packet will be corrupted and dropped by the gateway. On the other hand, in pure ALOHA without such global synchronization and discrete times, transmission times are continuous and random, lowering the probability that the preamble and header interfere with other transmissions and positively affecting scalability.

As a consequence, slotted ALOHA will not greatly increase scalability and more advanced scheduling techniques need to be applied where communication slots are assigned to end nodes. Rizzi *et al.* [22] showed how one can boost up the LoRaWAN network scalability by introducing a TSCH-like scheduling system for LoRa. However, in case the slot length encompasses the maximum packet transmission duration as well as both receive windows, much of the capacity will be lost due to receive windows that might never be used.

Therefore, we propose a fine-grained synchronization and scheduling solution for LoRaWAN networks where an network synchronization and scheduling entity (NSSE) resides at the LoRaWAN network server. Considering the fact that for LoRaWAN class A devices, downlink communication is always triggered by uplink traffic, our synchronization and scheduling method must be triggered by the end nodes. By means of infrequent signaling messages between the end node to the NSSE, and triggered by the end node, the end node can become synchronized and get communication slots assigned. The scheduling of communication slots by the NSSE is based on several end node parameters such as uplink traffic update rate, clock drift accuracy, and/or resynchronization frequency. In order not to break the LoRaWAN MAC standard, all signaling messages run on top of the LoRaWAN MAC layer.

As an alternative one could consider beaconing-based solutions, where gateways generate beacons toward all end nodes. However, this requires all nodes to listen for such beacons, conflicting with the behavior of class A LoRaWAN devices and increasing the energy consumption. Further, the size of beacon packets is limited, so including scheduling information inside beacons and doing this for large number of nodes, would either require very large packets or very frequent beacons toward subsets of nodes. The latter again conflict with the duty cycle limitations gateways must adhere to. Last but not least, in order for a gateway to serve all nodes with synchronization beacons, only SF12 can be used due to the longer coverage. This result in long time on air and longer waiting times between each transmission. Contrary, if lower SFs are used for beaconing, the synchronization coverage zone will decrease. Considering all the above limitations regarding beaconing-based solutions, we can motivate our decision to design the proposed synchronization and scheduling process as an active process, triggered by the end nodes.

As the synchronization is an active process, it might have an impact on data transmission if it happens in-band, i.e., using the same channel that is being used for scheduled data transmissions. Unsynchronized nodes can interfere with already synchronized nodes during transmissions of their initial synchronization requests. If the initial synchronization happens out-of-band this will be alleviated. The next limitation is the half-duplex property of gateways. If at a certain time, an uplink transmission is scheduled for an end node, the gateway has to listen to this translation and cannot simultaneously reply to synchronization requests even when happening in a different channel. Once synchronized, further resynchronization can be scheduled in advance by the network and can happen in the same band as the data traffic without impacting that traffic. So, the proposed solution foresees different combinations of the in-band and out-of-band synchronization and resynchronization policies. In Fig. 1, the synchronization packet flow is shown. In the following section, we discuss in further detail the designed solution.

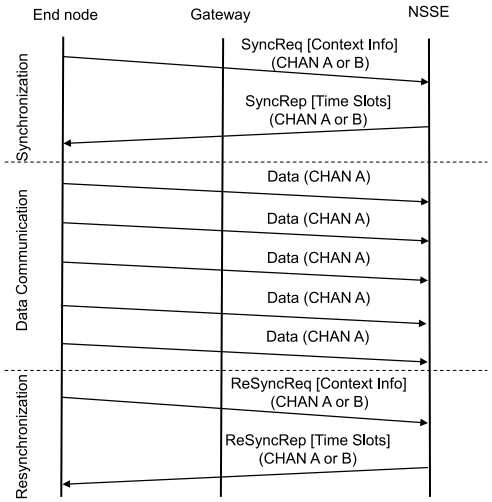


Fig. 1. Generic synchronization packet flow over time.

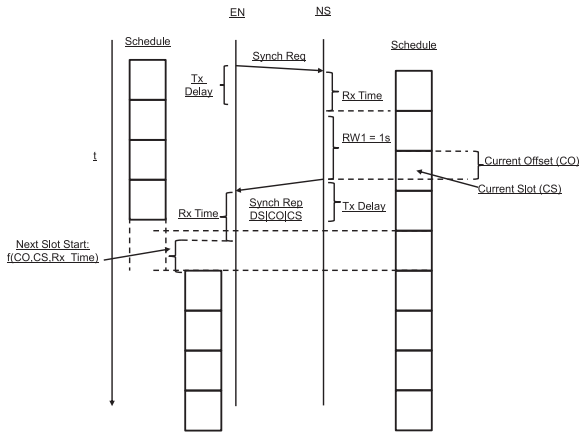


Fig. 2. Synchronization process between end node and network synchronization and scheduling entity.

V. Infrequent Synchronization and Fine-Grained Scheduling

A. End Node and NSSE Signaling

The NSSE is a central scheduler for the LoRaWAN network that schedules all uplink and downlink transmission for end nodes. Time is divided in time slots that can accommodate the longest transmission in the network plus the guard time. The guard time is calculated based on the maximal synchronization period in the network and the clock accuracy of end nodes. Each time slot is identified by its time slot index, which increases with time.

At the end node side, on top of the LoRaWAN MAC layer, there is a synchronization component that is responsible for generating synchronization requests and processing synchronization replies. Inside a synchronization request, the end node can include information such as the requested uplink traffic periodicity, clock accuracy, and/or resynchronization periodicity. Once the synchronization request is received by the NSSE, the information is processed and the lookup of available time slots for the end node can start. The time slot assignment is based on the availability of time slots, the information that is included in the synchronization request, the SNR and channel at which the synchronization request has been received. A time slot is assigned to an end node only if:

1) the time slot is available (not used);
2) an already assigned end node on that time slot uses a different channel than the requesting node;
3) the assignment of that slot to the requesting node does not decrease the SNR for already assigned nodes and for the node itself considering an acceptable threshold.

Each assigned node in a slot will be seen as a source of interference for other co-assigned nodes. Thus, the last condition makes sure that the newly assigned slot will not degrade the communication quality for other already assigned nodes in that slot.

The number of assigned time slots to an end node depends on the information provided by the end node in the synchronization request packet. If the end node provides the required synchronization and traffic periodicity, the number of assigned time slots will be $S_p \div T_p$, T_p, and S_p being traffic and synchronization period, respectively. If only clock accuracy and traffic periodicity are provided, a set of time slots is assigned to the node, after which the node must resynchronize. This assures that an end node does not interfere with other end nodes due to the clock inaccuracy.

In the reply back to the end node, the NSSE at the network server includes the current time slot index, the time offset in the current time slot and the future time slots indexes during which the end node is allowed to transmit. As the number of assigned time slots can be high and time indexes large because of the infrequent synchronization, the time slot indexes cannot be transmitted as a raw sequence of indexes due to packet length limitations. Rather, the assigned time slots are communicated to the end node using a space-efficient probabilistic data structure (Bloom filters) [23], as is explained in Section V-B. This data structure always produces a fixed length bit array that probabilistically represents whether a time slot is part of that data structure or not.

Based on the current time slot index and time offset in the current time slot information contained in the reply packet, as well as knowing the SF that was used in downlink and which receive window, the end node determines the current point in time as seen from the network. From that moment, the end node starts increasing the time slot index every TS_{length}, which is assumed to be known by every node in the network. This is shown in Fig. 2.

Once the synchronization has been performed and slots have been assigned to the end node, a scheduling component in the end node can check for every increment of the time slot index, whether that index is part of the probabilistic data structure. If so, the node can schedule a data transmission in the next slot. In order to save energy and decrease processing, the checking of the data structure can be done according to the traffic periodicity, i.e., in a time window following the next planned transmission. This will also avoid false positives outside that time window, as Bloom filters are associated with false positives.

B. Time Slot Assignment and Retrieval Using Bloom Filters

Fine-grained scheduling on a per node basis is not possible without two main mechanisms: 1) a mechanism to assign time slots to individual end nodes by the NSSE and 2) a mechanism for an end node to find which slots have been assigned.

For each synchronization request, the NSSE at the network server will decide to answer or not based on the already scheduled data transmissions for other end nodes in uplink and the duty cycle limitations of the gateways in downlink. As LoRa gateways are half duplex, the NSSE will only reply to a synchronization request in case the reply does not interrupt any scheduled uplink transmission. Based on the uplink traffic requirements and resynchronization periodicity requested by the end node, the NSSE will look up available time slots. Time slots will be filled in a "first come first serve" manner. This means that when a request is received by the NSSE, it checks whether the time slot after the requested traffic periodicity is available. If it is available, it will be assigned to that node, and the NSSE will continue to look for a slot in the next period. If it is not available, it will search for the first time slot that is available in positive direction from the start of the traffic period. This is shown in Fig. 3. So, as a consequence, the scheduled time slot can only be located at the requested traffic periodicity time or in its positive time direction. This rule makes it possible for the end node to only start checking for scheduled time slots at the beginning of every new traffic period and guarantees that no assigned time slots are missed.

If a time slot is being assigned, it will be added to the space-efficient data structure. The lookup for available time slots will continue until $n * T_p > S_p$, where n is increased every time a time slot for a traffic period is added to the data structure. It can happen that for a certain number of end devices and traffic requests, the full capacity of the network is reached. If this is the case, the NSSE will no longer reply to new synchronization requests. If the synchronization reply is not received, the end node will defer sending requests for a time. End nodes will send synchronization request with a 0.1% duty cycle, similar to the join request duty cycle in LoRaWAN standard [8]. This is done in order not to overflow the network with synchronization requests once the maximum capacity has been reached. The complete algorithm for assigning time slots to requesting nodes is given in Fig. 4.

In our current implementation, Bloom filters [23] are used for the realization of the space-efficient probabilistic time slot data structures. Bloom filters are used to check whether an element is part of the data structure or not. There can be false positives, but no false negatives, meaning that one time slot is either "not assigned" to the node or is "probably assigned." Two parameters are crucial for Bloom filters: 1) the filter size in bits and 2) the number of hash functions being used. The false positive probability of a Bloom filter, p, is determined by the number of entries in the data structure, n, the filter size in bits, m, and the number of hash functions used, k, and given by the following expression [24]:

$$p \approx \left(1 - e^{-kn/m}\right)^k. \tag{1}$$

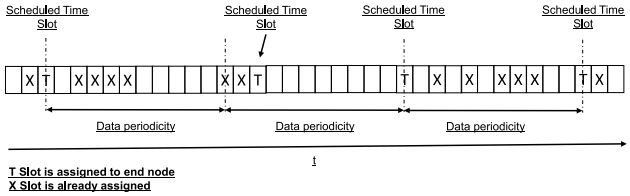


Fig. 3. Finding available time slots by the NSSE, considering the traffic periodicity of the end node.

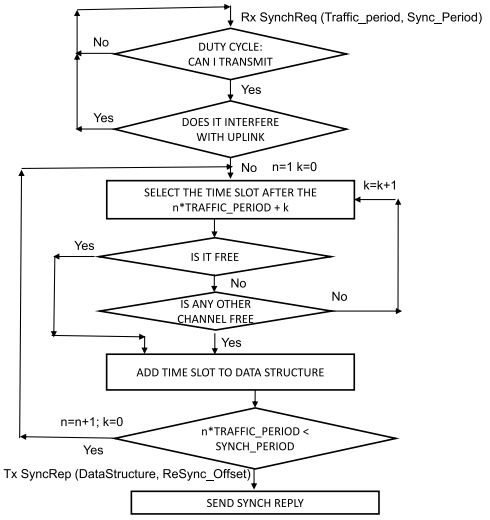


Fig. 4. Algorithm at the NSSE for assigning time slots to end nodes.

In order to speed up the hash function calculation, we use the double hashing technique [25], where all k hash values are calculated using only two hash functions

$$\text{hash}_i(x, m) = (\text{hash}_a(x) + i \times \text{hash}_b(x))\%m \tag{2}$$

with m the Bloom filter size, hash_a and hash_b two hash functions and i an ordinate and % modulus operation.

The data structure then provides all time slots until the next resynchronization period. This information is communicated by the NSSE to the end node, together with the next resynchronization time slot, expressed as a time slot offset to the current time slot.

In order to decrease the false positive probability for neighboring entries, cryptographic hash functions can be used provided they are not too time consuming and power hungry. Cryptographic hash functions have high avalanche effects, producing totally nonsimilar hash outputs for similar hash functions inputs. This is beneficial in case of the periodic checkups for increasing slot indexes at the end node side.

At the end node side, the reverse procedure of Bloom filtering needs to be done. We assume all hash functions that are used to add entries to the data structure are known to all nodes in the network. The end node needs to pass the time slot index to be checked to these hash functions. The output value, a bit vector, is compared with the bit values inside the data structure. If the set bit positions are different from those inside the data structure, the time slot is certainly not part of the data structure and the end node cannot transmit in that

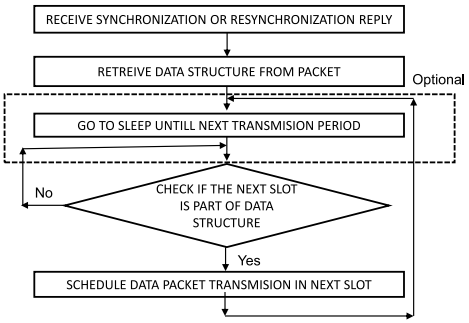


Fig. 5. Algorithm for checking if time slot is part of data structure in the end node. Optional part can be removed for continuous check up.

time slot. As it is shown in Fig. 3, the assigned time slot is always assigned at the beginning of the requested traffic periodicity or in its positive time direction. So, the node can start checking if the time slot is part of the data structure on every data traffic period. If the first time slot is not part of it, the end node can move on to the next slot and continue until a slot is found. Once found, the end node can defer checking for other time slots until the next data traffic period. As such, the number of checkups will reduce and false positive outside these time windows will not occur. In Fig. 5, the algorithm to check for the assignment of time slots at the end node side is given.

VI. Mathematical Optimality Problem

In single gateway networks there is a tradeoff between traffic frequency and synchronization frequency. The time slot length must be increased with increasing synchronization periods as it has to take into account the clock drift of end nodes within one synchronization period. This in turn result in a lower amount of time slots for data traffic. On the other hand, the number of end nodes that can be served with synchronization replies by a single gateway will increase with an increasing synchronization period. So, for a given traffic periodicity and clock drift and assuming this is the same for all nodes, there will be an optimal synchronization frequency, where the number of available time slots within one traffic period will be equal or lower than the number of end nodes that can be served with synchronization replies by the gateway within one synchronization period.

Let f_1 be the function that describes the number of time slots within one traffic period as a function of the synchronization period. Let f_2 be the function that describes the relation between the number of end nodes that can be served within one synchronization period by a gateway as a function of that synchronization period. While function f_1 decreases monotonically with an increasing synchronization period, function f_2 increases monotonically with it. The optimal synchronization period will be the period that fulfills the condition

$$f_1(S_P) <= f_2(S_P) | S_P = S_P\text{Opt} \tag{3}$$

where S_POpt is the optimal synchronization period.

TABLE I
Simulation Parameters

Parameter	Value
Data packet length	21 B
Sync REQ length	15 B
Sync REP length	28 B
Resync Period	1 day
Traffic Period	10 min
Tx power	14 dBm
Simulated Time	25 h
Cell Radius[1] (SF12,SF11,SF10,SF9,SF8,SF7)	(6100,4500,3600,3050,2600,2300) m
Time Slot Length [2] (SF12,SF11,SF10,SF9,SF8,SF7)	(2.67,1.88,1.37,1.14,1.02,0.94) s

[1]Coverage determined using LogDistancePropagationLoss model for a PER of 0.01. See [16]. [2]10ppm clock drift for 24h is considered too.

Let T_P be the traffic periodicity and TS_{length} the time slot length. The function f_1 can be written as

$$f_1(S_P) = \frac{T_P}{TS_{\text{length}}}. \tag{4}$$

Let A_t be the air time of the synchronization reply and W_t the waiting time between two consecutive synchronization replies by the gateway. The function f_2 can be written as function of synchronization period as

$$f_2(S_P) = \frac{S_P}{A_t + W_t}. \tag{5}$$

All the right parts of (4) and (5) are functions of known variables and S_P. Further, $TS_{\text{length}} = A_{t_{\text{Max}}} + Cl_{\text{drift}} * S_P$, where $A_{t_{\text{Max}}}$ is the air time of the largest packet that can be transmitted and Cl_{drift} is the clock drift of the end node, both known variables. Thus, for a fixed T_P, the optimal synchronization period can be calculated using (3) for S_P.

VII. Results

To validate our approach, we implemented the proposed synchronization scheme on top of our LoRaWAN simulator [16], [26] in NS-3. The synchronization and scheduling components run on top of the LoRaWAN MAC layer together with the application layer on both ends, i.e., the end device and network server. The NSSE at the network server keeps track of the schedule by updating the time slot index. Each time slot assignment that is made by the NSSE, is saved in a data structure that includes the channel at which the communication happens, the node ID to which the time slot is assigned and the SNR value with which the synchronization request was received. These data structures are saved in an unordered hash map with the time slot index as key word. For memory efficiency, after every time slot index increment, all data structures of previous time slots will be cleared from the map. In the current end node implementation, the end node sends the synchronization request, thereby asking for a certain data traffic period. Further, the implemented mechanism ensures the synchronization duty cycle to be less than 0.1%.

In the following sections, we show the results in terms of network PDRs for an increasing number of devices in

the network. We also show the total number of data packets successfully delivered over the network. In this set of simulations, the time slot assignment is done based on the requested traffic periodicity by the end nodes. The synchronization periodicity is set to one day, while the time slot length is fixed to accommodate the longest transmitted packet. The implemented Bloom filter uses a filter length of 64 bits and six hash functions. We used the Murmur hash function [27] to calculate the hash values for the Bloom filter entries together with double hashing for speeding up hash calculations. The Murmur function has a low avalanche effect, meaning that similar inputs have similar output hashes. This will increase hash similarities for neighboring time slots and thus false positives. However, as it will be shown, the impact of false positives when using the Murmur function is still sufficiently low, while its implementation simplicity determined our choice. The data packet size used was 21 bytes and no downlink data traffic was considered except for scheduled in-band resynchronization replies. The rest of the simulation parameters are described in Table I.

End nodes are uniformly distributed in the cell. The cell radius coverage is determined using a log distance propagation model for a PER of 0.01% for each SF. The traffic model for each end node is assumed to be periodic with a uniform distribution of transmissions by end nodes in the first data period. As such, transmissions of nodes are randomized within the data period. All subsequent transmissions are determined based on the data period and potential clock drift over time. The clock drift is modeled as uniformly random between $[-\text{drift}, +\text{drift}]$, where drift is the maximal drift that can happen during the time period until next transmission. This makes it possible for the device to start a transmission even before the exact time period if it experiences a negative clock drift. This is realized by calculating the exact time when the end node has to transmit $\pm$ the drift drawn by the random process. A 10-ppm clock drift for each end node is considered. For the synchronized case, the length of the time slots was set in order to account for the time-on-air of the packet as well as the clock drift that can occur during the maximal synchronization periodicity and for a clock accuracy of 10 ppm.

Simulations are done using multiple data channels. Three 125-kHz channels at 868 MHz are considered. In the first case, the synchronization traffic uses the same channels as the data traffic (in-band synchronization) while in the second case the synchronization traffic uses the high power (20 dBm) high duty cycle (10%) channel at 868.52 MHz (out-of-band synchronization). In the second case, the impact of yet unsynchronized nodes on the already synchronized nodes will be alleviated and be practically zero. To be able to evaluate the proposed solution under saturated conditions, i.e., more uplink traffic demand than can be handled by the available network capacity, we opt to let end nodes transmit every 10 min and assess the network performance during a 1-h period. This enables us to limit the execution time of the simulations as saturation can be achieved with less nodes. However, in order to also consider the impact of clock drift on the scheduling mechanism, i.e., slots must incorporate a guard time, we consider a warm-up period of one day during which nodes are being added to the network and get synchronized. This warm-up period is then followed by the 1-h data transmission period. So, worst case, an end node starts at the 1-h data transmission period with a clock drift built up during one day. We run the simulations on the imec virtual wall testbed [28]. Even with the above approach, a single simulation could go up to 12 h even on a server with a 2× Quad core 2.2-GHz CPU and 12-GB RAM.

TABLE II
SIMULATION PARAMETERS

SF	Number of time slots in one data period of 10 min.	Total number of end nodes that can be supported.[1]
SF12	224	672
SF11	318	954
SF10	436	1308
SF9	522	1566
SF8	588	1764
SF7	632	1896

[1]Time slots that are needed to be used for resynchronization are not considered in this table as that one is case specific depending how many end nodes are scheduled to be resynchronized.

In Table II, the maximal number of time slots in one data period is shown. The maximal number of supported end nodes will be approximately three times higher as we are using three different channels. These numbers do not include the time slots reserved for resynchronization as they are case specific, depending on how many nodes need to be resynchronized within one data traffic interval. The simulated range of end devices is selected based on Table II. This range accounts for the maximal number of supported end devices per data period as well as some cases where the range is higher to see the impact in both synchronized and unsynchronized case. For these numbers of end devices, there are no limitations with respect to the amount of synchronization replies from the gateway (due to duty cycle) as the synchronization period is sufficiently long. In this case, the gateway can send from $\sim$ 5300 synchronization packets in one day for SF12 up to $\sim$ 112 609 in case of SF7. These numbers are much higher than the total number of end devices that can be scheduled with the given traffic periodicity.

A. In-Band Synchronization

In case of in-band synchronization, all three channels are used for data and synchronization information exchange. As there is no separate channel for synchronization and the synchronization process is an active process, unsynchronized nodes may interfere with the data traffic of already synchronized nodes. In this case, we expect a lower utilization of the available capacity by end nodes and higher losses than the out-of-band synchronization case. The number of end devices in the network is based on the theoretical number of supported devices in each case, as it is given in Table II, thereby considering a range from below to above the theoretical maximum supported end devices.

In networks with a low number of end devices and where synchronization frequencies are low, it is worthy not to "waste" a channel by reserving it only for synchronization information

(at least it should be used for both synchronization and normal random access traffic, i.e., nonscheduled traffic). The PDR behavior depends on the SF. As it is shown in Fig. 6, the synchronization method achieves a better PDR than the unsynchronized case for SF12 to SF9 (note that only SF12 results are included in the figure), while for SF7 and SF8 (note that only SF7 results are included in the figure), the PDRs for both cases are similar. For the synchronized case, the decrease in PDR is due to false positives of the Bloom filter in case the number of end devices is lower than the theoretical maximum. In case the number of end devices becomes higher than the theoretical maximum, part of the losses is caused by unsynchronized nodes that will continue sending synchronization request in the same channels as of data traffic. However, this impact is lower than the one caused by data traffic collisions in the unsynchronized case as it uses only 0.1% duty cycling compared with 1% for data traffic. For each SF, the synchronized solution achieves PDRs that are above 90%.

Regarding the total number of delivered packets, the synchronized case outperforms the unsynchronized case too. However, the total number of delivered data packets is lower than when the synchronization happens out-of-band. In case of out-of-band synchronization, the NSSE needs to reserve only one time slot per end device for sending synchronization replies due to half duplex gateways. There is no need to reserve any time slot for resynchronization requests as these happen in a different channel and will not consume slots otherwise used for data traffic. Contrary, in case of in-band synchronization, as synchronization happens in the same channels as data traffic, the NSSE needs to reserve one time slot for the synchronization request and another one for the synchronization reply, decreasing the number of time slots available for data traffic. This will decrease the amount of delivered data packets compared to out-of-band synchronization. This can be noticed by comparing the curves of total delivery data packets for synchronized case between Figs. 6 and 7 for each case.

B. Out-of-Band Synchronization

In Fig. 7, the PDR only for SF12 and SF7 and different number of end nodes in the network is given for out-of-band synchronization. The number of simulated end devices is determined based on Table II, again considering a range from below to above the theoretical maximum supported end devices. For all SFs, the synchronization method outperforms the unsynchronized method in terms of PDR (in Fig. 7), being 7% (for SF7) up to 30% (for SF12) higher than the unsynchronized case. The actual PDR for the synchronization case is higher than 98% in all cases except in case of SF7. The losses are mainly due to false positives of the Bloom filter as well as low SNR losses related with peripheral end devices. As we determined the cell radius based on 1% packet error rates of the network [16], nodes at the periphery of the cell experience higher packet losses due to worse SNR conditions. Moreover, when the number of end nodes is increased the number of peripheral nodes is increased too. These loses due to peripheral nodes will not contribute to more than 1% of the total losses in network.

In terms of the total number of delivered data packets, we achieve the maximal achievable figures. Once the total number of end devices is reached, e.g., in Fig. 7(b) cases of 1900–2000 end devices, the total number of delivered data packet stays steady while the PDR remains the same. This is due to the fact that an increasing number of end devices will not impact the data traffic of the synchronized end devices. The reverse is valid for the unsynchronized case. Once the number of end devices is increased, the data PDR will continue to decrease as well as the total number of delivered data packets [e.g., Fig. 7(b) for SF7]. Note that in case when the number of end devices is higher than the maximum theoretical one, a number of end devices will not transmit any data packet as they will not get synchronized due to the lack of capacity. This number is the difference of theoretical maximum number of end devices from the actual number of end devices. Fig. 7 shows results only for SF12 and SF7. Note that the same behavior is observed for other SFs too.

a) Additional considerations: Calculating the data points shown in Figs. 6 and 7 is a compute intensive process and takes up to 12 h for a single simulation run even on powerful machines (2× Quad core 2.2-GHz CPU and 12-GB RAM). As such, running every simulation many times in order to produce confidence intervals is very time consuming due to the extensive simulation times. Nevertheless, in order to get an idea of the accuracy of the obtained results, we selected the scenario for 500 end nodes and SF12 and ran that simulation 50 times for both synchronized and unsynchronized case. In case of in-band synchronization, the average PDR across all simulations was 0.96, with maximal and minimal values of 0.97 and 0.95, respectively. For the case of out-band synchronization, we obtained an average of 0.986 and min–max values of 0.981 and 0.991, respectively. For the unsynchronized case with the same settings the average PDR value was 0.66 and min–max values of 0.64 and 0.67, respectively. It can be seen that the min–max values do not deviate that much from the average. This can be explained because each simulation already involves quite some randomness including the end node distribution model in the cell, the traffic model and the clock drift model, and the PDR value shown is the value obtained by averaging the individual PDR values for a large number of nodes.

To further assess the performance improvement of our solution, we compare the worst case end node PDR experienced by a single node for the out-of-band synchronization case with both best case end node PDR experienced by a node and average PDR over all nodes for the unsynchronized case. This is shown in Table III. This is an extreme comparison that gives more insights in the improvements that the proposed synchronization scheme brings. It can be seen that in case of SF12–SF10 the best case end node PDR for the unsynchronized case will not even pass the worst case end node PDR for the out-of-band synchronization case. In addition to this, the average PDR in the unsynchronized case is lower than the worst case node PDR in the out-of-band synchronization case for SF12 to SF9. In case of SF8 and SF7, the average PDR for the unsynchronized case is only 0.06 higher than the worst case node PDR in out-of-band synchronization case, but

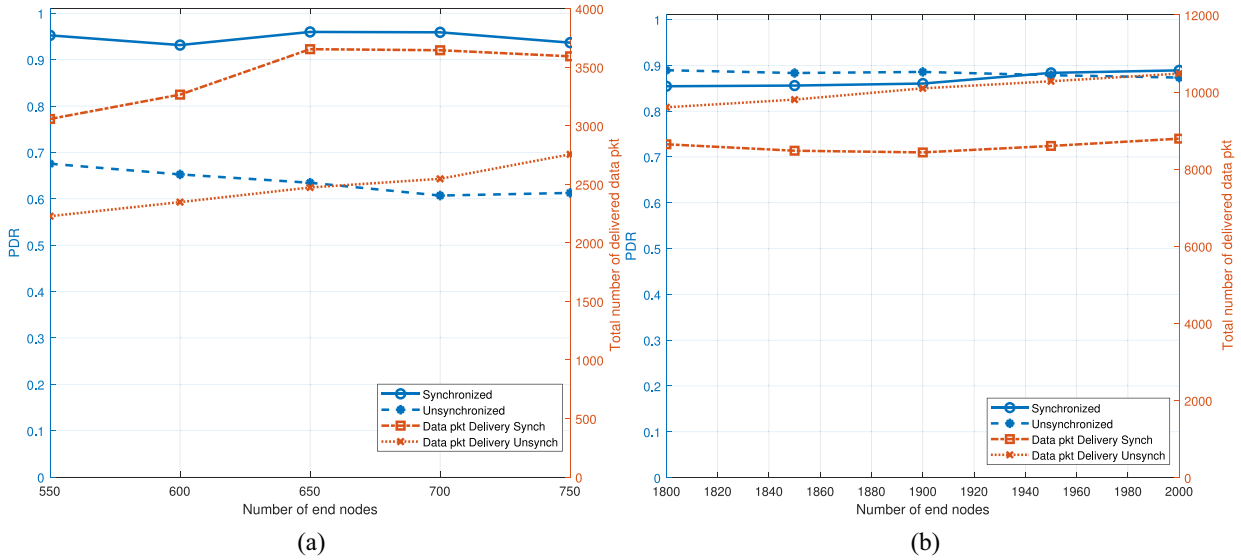


Fig. 6. PDR and the total number of delivered data packets for synchronized and unsynchronized case using multiple channels. Synchronization is done in the same channels as the data traffic channels. Similar behavior is observed for other SFs too. (a) SF 12, data rate 0. (b) SF 7, data rate 5.

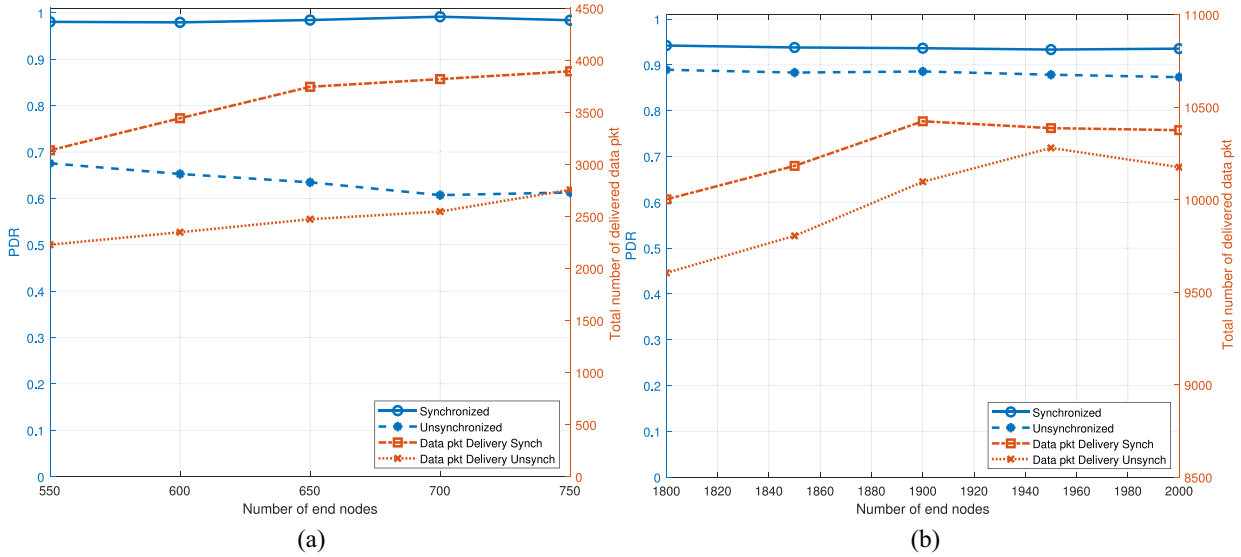


Fig. 7. PDR and the total number of delivered data packets for synchronized and unsynchronized case using multiple channels. Synchronization is done in a different channel than the data traffic channels. Similar behavior is observed for other SFs too. (a) SF 12, data rate 0. (b) SF 7, data rate 5.

of course lower than the average PDR for the unsynchronized case as shown in Fig. 7 (for SF7).

Currently, ALOHA is used for massive scale IoT where periodic monitoring traffic is used. The scope of the proposed synchronization solution targets such cases as their traffic organization can be achieved. The proposed solution can also cover cases with event-based traffic where a certain delay can be tolerated. For use cases where low delay event-based traffic has to be supported, the proposed synchronization scheme is not suitable and has to be complemented with a mechanism for sending event-based traffic out-of-band in a similar way as synchronization request are sent. Further, in addition of providing a separate channel for such traffic, a dedicated SF can be used for low delay event-based traffic. Such a solution will offer lower packet error rates for such event-based transmissions compared to ALOHA, as they will share the channel only with synchronization requests, which are sent at a much lower duty cycle compared to what the real data traffic duty cycle can be (1%).

As LoRaWAN networks operate in unlicensed spectrum, there is always a risk that other nodes from other networks can transmit and interfere with the scheduled traffic. The interference effect will depend on the RSSI values seen at the receiver and time shift of the interferer from the start of the main transmission [15]. Nevertheless, the proposed synchronization solution can be applied also to cross-LoRaWAN networks as well as cross-technology networks.

C. Throughput Comparison

In this section, we present a theoretical comparison of the achievable network throughput between the unsynchronized

TABLE III
WORST CASE PDR EXPERIENCED BY A SINGLE NODE FOR OUT-OF-BAND SYNCHRONIZATION VERSUS THE BEST CASE AND AVERAGE PDR FOR THE UNSYNCHRONIZED CASE

SF12				SF11				SF10			
# end devices	Worst PDR Synch	Best PDR UnSynch	Avg PDR UnSynch	# end devices	Worst PDR Synch	Best PDR UnSynch	Avg PDR UnSynch	# end devices	Worst PDR Synch	Best PDR UnSynch	Avg PDR UnSynch
550	0.83	0.83	0.69	850	0.83	0.83	0.73	1250	0.83	0.83	0.77
600	0.83	0.83	0.65	900	0.83	0.83	0.70	1300	0.83	0.83	0.78
650	0.83	0.83	0.63	950	0.83	0.83	0.69	1350	0.83	0.83	0.78
700	0.83	0.66	0.61	1000	0.83	0.83	0.69	1400	0.83	0.83	0.73
750	0.83	0.66	0.60	1050	0.83	0.83	0.68	1450	0.83	0.83	0.73
SF9				SF8				SF7			
# end devices	Worst PDR Synch	Best PDR UnSynch	Avg PDR UnSynch	# end devices	Worst PDR Synch	Best PDR UnSynch	Avg PDR UnSynch	# end devices	Worst PDR Synch	Best PDR UnSynch	Avg PDR UnSynch
1450	0.83	1	0.84	1650	0.83	1	0.88	1800	0.83	1	0.9
1500	0.83	1	0.82	1700	0.83	1	0.87	1850	0.83	1	0.89
1550	0.83	1	0.81	1750	0.83	1	0.88	1900	0.83	1	0.89
1600	0.83	0.83	0.81	1800	0.83	1	0.87	1950	0.83	1	0.89
1650	0.83	0.83	0.81	1850	0.83	1	0.87	2000	0.83	1	0.88

case and synchronized case. For simplicity, we model unsynchronized LoRa communication as ALOHA-based communication.

Let N be the total number of end nodes in the network. Each end node transmits every data period T_p, meaning that there will be N transmissions per time period on average. The duration of a transmission is called the frame time and we consider it to be equal to the time slot duration TS_{length}. Based on this, we can express G, the average transmission attempts per frame time as

$$G = \frac{N * TS_{\text{length}}}{T_p}. \tag{6}$$

The node transmissions during one data period are uniformly distributed. Therefore, the probability of a node transmitting in a certain subinterval, I, of the data period will be

$$P_{TX}[I] = \frac{I}{T_p} \tag{7}$$

while the probability that the node will not transmit in a certain subinterval, I, of the data period will be

$$P_{noTX}[I] = 1 - \frac{I}{T_p}. \tag{8}$$

In order for a packet to be received correctly by the network, there should be no other transmissions in two consecutive time frames, $2 * TS_{\text{length}}$. In other words, in the interval of $2 * TS_{\text{length}}$. there should be one and only one transmission. The probability that there will be one and only one transmission in one subinterval, I, of data period will be

$$P_{1TX}[I] = \frac{I}{T_p}\left(1 - \frac{I}{T_p}\right)^{(N-1)} \tag{9}$$

where N is the total number of end nodes.

In case of ALOHA, normalized throughput, will be

$$\begin{aligned} Th &= G * P_{1TX}\left[2 * TS_{\text{length}}\right] \\ &= \frac{N * TS_{\text{length}}}{T_p} * \frac{2 * TS_{\text{length}}}{T_p}\left(1 - \frac{2 * TS_{\text{length}}}{T_p}\right)^{(N-1)}. \end{aligned} \tag{10}$$

Equation (10) gives the throughput per frame time when pure ALOHA is assumed. Considering now our out-of-band synchronization solution, and assuming no false positives, the probability that there will be only one transmission per time slot (frame time) is always 1. As such, the throughput can be expressed as

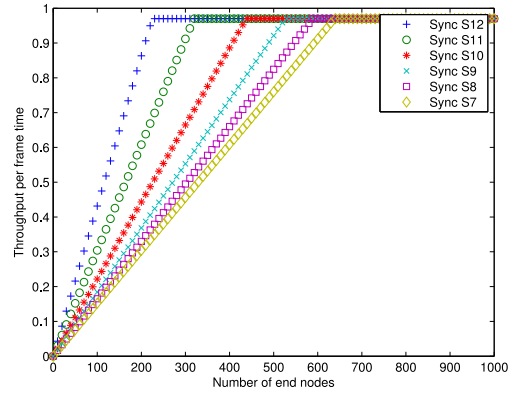
$$Th = G = \frac{N}{T_p}. \tag{11}$$

This formula gives the relation between the throughput and the number of end devices and the data transmission period of each end device. The maximal throughput is reached when N becomes equal to T_p/TS_{length}, after which it will remain constant as no new nodes can be scheduled. If we also take into account the false positive rate of the Bloom filter, (11) becomes

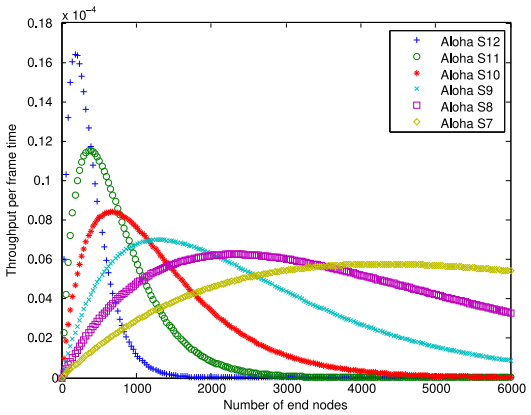
$$Th = (1 - p)\frac{N}{T_p} \tag{12}$$

where p is the false positive probability.

In Fig. 8, the average throughput per frame time normalized by the maximum throughput considering a false positives rate of 3% is shown, for each SF and for both synchronized and unsynchronized cases. The graphs are drawn using the above formulas, the time slot lengths from Table I and a traffic period of 600 s. In case of ALOHA [Fig. 8(b)] no guard time was considered, only the frame time. It can be seen that the maximal normalized throughput for the synchronized case [Fig. 8(a)] is always near 1 when the maximal number of end devices that can be supported is reached, e.g., 224 end devices for SF12 (see Table I). When the number of end devices becomes larger, the throughput remains constant at its maximal value, as unsynchronized nodes will not interfere with already synchronized nodes. On the other hand, the throughput for the unsynchronized case [Fig. 8(b)] will continue to decrease once it reaches its maximum. In this case, increasing the number of end nodes will saturate the network and traffic will experience a higher loss rate. It can be noticed that the maximal throughput in case of ALOHA is reached for higher number of end nodes compared with the synchronized case. This comes as a fact that no guard time interval should be considered for ALOHA case. However, ALOHA throughput per frame time



(a)



(b)

Fig. 8. Average throughput per time frame. The time slot length for each SF is taken from Table I while Tp = 600 s. Only one data channel is considered here and for the synchronized case a false positive rate of 3% is taken into account for the Bloom filters. (a) Synchronization out-of-band. (b) ALOHA.

is much lower than throughput in synchronized case. We are aware that the ALOHA model is an underestimation of the real achievable capacity, but it helps to show the trend of throughput decrease for an increasing number of end nodes in the network.

Fig. 8 showed the normalized throughput as a function of the number of end devices, using formulas (10) and (12), respectively. Here, the frame time still assumes the use of a guard time in order to account for the clock drift. This means that under perfect conditions, i.e., no clock drift and slots fully occupied by the data transmission only, the maximal theoretically throughput will be significantly higher. In addition, LoRaWAN packets have some packet header overhead, only part of the packet being used for the actual application data. Both aspects have been considered in Fig. 9, where the maximum theoretically achievable goodput (application layer data throughput) is plotted and compared against the synchronized solution for different synchronization periods and thus guard times and the unsynchronized case. In this case, a clock drift of 10 ppm is assumed for the synchronization case, while for the unsynchronized case the ALOHA model is used. It can be seen that for lower SFs, the synchronization case has a lower goodput due to the higher ratio between the guard

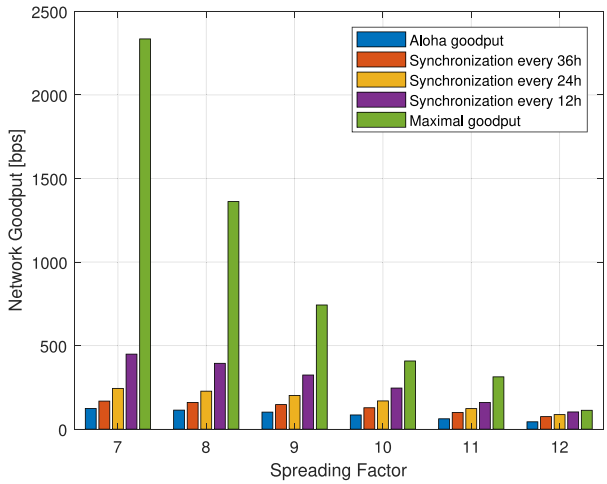


Fig. 9. Throughput comparison between LoRa case (modeled as ALOHA), synchronized case for different synchronization frequency and maximal achievable throughput.

time and the actual duration of the packet. Also differences can be seen in goodputs for different synchronization periods. For lower synchronization periods the guard time for clock drift can be kept smaller, resulting in a higher achievable goodput. However, in any case the goodput for the synchronization case is higher than the goodput for the unsynchronized case.

D. Battery Capacity Usage Overhead

Performing synchronization and scheduling comes at a signaling cost and thus additional power consumption. As all the end nodes are battery-powered it is relevant to evaluate the impact of the synchronization overhead on the battery lifetime. The overhead for the synchronization case will consist of sending one synchronization request and receiving one synchronization reply packet every synchronization period. For a number of end devices that is lower than the maximum number of end devices that can be served within one data period, the synchronization overhead is only one packet per synchronization period. In case of a synchronization period of 1 h and a data period of 10 min, one synchronization packet for every six data packets is needed, resulting in a ∼16% traffic overhead. By increasing the synchronization period, this overhead can be decreased at the expense of a larger Bloom filter data structure in order to keep the false positive probability low. By doing so, the better PDR will compensate the additional signaling overhead, as in the unsynchronized case, energy is spent on sending packets that will get lost due to collisions and that might require retransmissions.

In case the number of end devices becomes higher than the maximum number of end devices that can be supported, the average synchronization overhead per device will be higher. However, different mechanisms can be applied in such cases, e.g., synchronization duty cycling (already implemented in current solution), preventing nodes transmitting more than a certain number of synchronization packets in a time period,

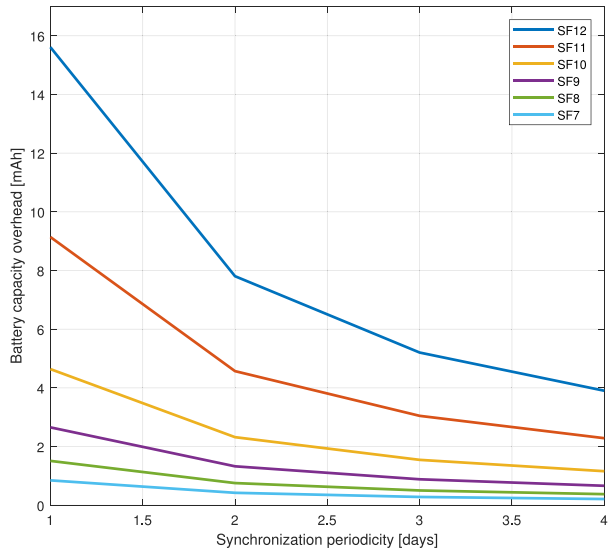


Fig. 10. Synchronization overhead in battery capacity usage for different SFs and different synchronization periodicity during a period of one year.

or even precalculation of the maximum supported number of end devices.

To calculate the impact of the synchronization approach on the battery lifetime, we consider cases where the number of end devices is lower than the theoretical maximum. For these calculations, we assume an SX1272 [29] LoRa chip for the end devices. According to [29], a LoRa module will use 10.5 mA in receive mode and 90 mA in transmit mode. Taking into account that each node will send one synchronization request of 15 bytes and will receive a synchronization reply of 28 bytes we calculate the additional battery capacity consumption due to synchronization and scheduling over a period of one year.

In Fig. 10, the synchronization overhead in terms of battery capacity usage is given for a period of one year. It can be seen that by increasing the synchronization periodicity to more than three days, the synchronization process will require less than 3 mAh extra over one year period, except for SF12. For SF12, a synchronization periodicity greater than four days will decrease the battery capacity usage overhead to less than 3 mAh. In general, this battery capacity usage overhead is much lower than what a node will use to transmit packets that will never reach the network in the unsynchronized case, as it was shown based on the number of packets delivered to the network in Sections VII-A and VII-B. Of course, an optimal configuration must consider multiple aspects such as clock drift, traffic periodicity, energy cost of synchronization, size of Bloom filter data structure, etc.

VIII. Conclusion

This paper presents a fine-grained synchronization and scheduling mechanism for LoRaWAN class A devices that can be realized on top of the existing LoRaWAN MAC layer. The NSSE at the network server schedules uplink and downlink traffic for end nodes by means of a central scheduler. Each end node has to request time slots by contacting the NSSE before it will be able to transmit data packets. Such requests can be done either in-band or out-of-band. The NSSE uses a space-efficient data structure, namely Bloom filters, to encode the time slot indexes in order to reduce the synchronization packet length and to be able to send more info toward the end node. Time slots are assigned to end nodes based on their traffic needs and other context information such as synchronization periodicity, clock drifts, etc. The end node uses the same filtering structure to find whether a time slot is part of the data structure or not. The false positive probability will decrease even further when using cryptographic hash functions with high avalanche effect and limited time slot checking to limited time windows.

By means of simulations, we showed the resulting PDR and total number of delivered data packets of our solution. In both cases, in-band and out-of-band synchronization, the synchronized method outperforms unsynchronized communication in terms of PDR and total amount of delivered data packets. In case of in-band synchronization, the total number of delivered packets is lower than in case of out-of-band synchronization, as the synchronization requests-replies had to be scheduled in the same channels as the data traffic, loosing thus some capacity. For out-of-band synchronization, the PDR is 7% (for SF7) to 30% (for SF12) higher than for the unsynchronized case. For saturated networks, the differences in PDR become more profound as nodes are only scheduled as long as they can be accommodated given the remaining capacity of the network. In terms of synchronization overhead it was shown that for a synchronization period higher than three days, the synchronization process will require less than 3-mAh extra battery capacity over a one year period per end device. This is much lower than the battery capacity used by end nodes to transmit data packets that never arrive at their destination due to collisions.

To conclude, we can say that this paper has demonstrated the feasibility of performing fine-grained synchronization and scheduling in LoRaWAN networks, without requiring any modifications to the MAC layer. The current analysis only considered uniform periodic traffic as well as fixed time slots length for all end nodes. One can easily think of more complex scenarios with sparser and more heterogeneous traffic as well as more advanced and intelligent algorithms that can use different time slots length for end nodes. These aspects should be considered as interesting follow-up work to extend the current work. Also a multi-SF scheduling algorithm that takes into account inputs from the physical layer, e.g., SNR and RSSI values of the requesting node, can be considered as an interesting extension. At least, the implementation of such a synchronization procedure on real LoRaWAN devices can be one of the future steps, in order to validate the approach in practice.

References

[1] P. Middleton *et al.*, "Forecast: Internet of Things, endpoints and associated services, worldwide, 2014," Gartner, Stamford, CT, USA, Rep. G00347577, 2014.

[2] (May 2017). *Sigfox*. [Online]. Available: https://www.sigfox.com/

[3] J. W. E. H. D. Landstrom and S. Bergstrom. (May 2017). *NB-IoT: A Sustainable Technology for Connecting Billions of Devices*. [Online]. Available: https://www.ericsson.com/en/publications/ericsson-technology-review/archive/2016/nb-iot-a-sustainable-technology-for-connecting-billions-of-devices

[4] LoRaWAN. (Jan. 2018). *What is it? A Technical Overview of LoRa and LoRaWAN*. [Online]. Available: https://www.lora-alliance.org/

[5] (Jan. 2018). *Weightless*. [Online]. Available: http://www.weightless.org/

[6] O. B. Seller and N. Sornin, "Low power long range transmitter," U.S. Patent 9 252 834, Feb. 2, 2016.

[7] O. Georgiou and U. Raza, "Low power wide area network analysis: Can LoRa scale?" *IEEE Wireless Commun. Lett.*, vol. 6, no. 2, pp. 162–165, Apr. 2017.

[8] *LoRaWAN R1. 0 Open Standard Released for the IoT*, LoRa Alliance, Fremont, CA, USA, 2015.

[9] *Electromagnetic Compatibility and Radio Spectrum Matters (ERM); Short Range Devices (SRD); Radio Equipment to be Used in the 25 MHz to 1 000 Mhz Frequency Range With Power Levels Ranging Up to 500 MW, V2*, Eur. Harmonized Standard EN 300 220-1, 2012.

[10] *LoRaWAN Regional Parameters V1. 0*, LoRaWAN, Fremont, CA, USA, Jul. 2016.

[11] K. Mikhaylov, J. Petaejaejaervi, and T. Haenninen, "Analysis of capacity and scalability of the LoRa low power wide area network technology," in *Proc. Eur. Wireless 22nd Eur. Wireless Conf. (VDE)*, 2016, pp. 1–6.

[12] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley, "A study of LoRa: Long range & low power networks for the Internet of Things," *Sensors*, vol. 16, no. 9, p. 1466, 2016.

[13] F. Adelantado *et al.*, "Understanding the limits of LoRaWAN," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 34–40, Sep. 2017.

[14] D. Bankov, E. Khorov, and A. Lyakhov, "Mathematical model of LoRaWAN channel access," in *Proc. IEEE 18th Int. Symp. World Wireless Mobile Multimedia Netw. (WoWMoM)*, 2017, pp. 1–3.

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

[16] 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, Dec. 2017, doi: 10.1109/JIOT.2017.2768498.

[17] T. Voigt, M. Bor, U. Roedig, and J. Alonso, "Mitigating inter-network interference in LoRa networks," in *Proc. Int. Conf. Embedded Wireless Syst. Netw.*, 2017, pp. 323–328.

[18] B. Reynders, W. Meert, and S. Pollin, "Power and spreading factor control in low power wide area networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2017, pp. 1–6.

[19] F. Cuomo *et al.*, "EXPLoRa: Extending the performance of LoRa by suitable spreading factor allocations," in *Proc. IEEE Wireless Mobile Comput. Netw. Commun. (WiMob)*, 2017, pp. 1–8.

[20] M. Slabicki, G. Premsankar, and M. Di Francesco, "Adaptive configuration of LoRa networks for dense IoT deployments," in *Proc. NOMS IEEE/IFIP Netw. Oper. Manag. Symp.*, 2018, pp. 1–9.

[21] K. Q. Abdelfadeel, V. Cionca, and D. Pesch, "A fair adaptive data rate algorithm for LoRaWAN," *arXiv preprint arXiv:1801.00522*, 2018.

[22] M. Rizzi, P. Ferrari, A. Flammini, E. Sisinni, and M. Gidlund, "Using LoRa for industrial wireless networks," in *Proc. IEEE 13th Int. Workshop Factory Commun. Syst. (WFCS)*, 2017, pp. 1–4.

[23] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[24] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

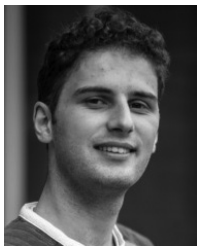
[25] A. Kirsch and M. Mitzenmacher, "Less hashing, same performance: Building a better bloom filter," *Random Struct. Algorithms*, vol. 33, no. 2, pp. 187–218, 2008.

[26] (Jan. 2018). *LoRaWAN NS3 Simulator*. [Online]. Available: https://github.com/imec-idlab/ns-3-dev-git/tree/lorawan

[27] A. Appleby. (2008). *Murmurhash*. [Online]. Available: https://sites.google.com/site/murmurhash

[28] (Jan. 2018). *IMEC Wilab Testbed*. [Online]. Available: http://doc.ilabt.iminds.be/ilabt-documentation/virtualwallfacility.html

[29] (Jan. 2018). *Sx1272/73–860 MHz to 1020 MHz Low Power Long Range Transceiver*. [Online]. Available: http://www.semtech.com/images/datasheet/sx1272.pdf



Jetmir Haxhibeqiri received the bachelor's degree in telecommunication from the University of Prishtina, Prishtina, Kosovo, in 2010, and the master's degree in engineering (information technology and computer engineering) from RWTH Aachen University, Aachen, Germany, in 2013. He is currently pursuing the Ph.D. degree at the Internet and Data Laboratory Research Group, Department of Information Technology, Ghent University, Ghent, Belgium.

During his studies, he had the opportunity to do internships in Deutche Telekom (T-Laboratories), Darmstadt, Germany. During this placement, he was a part of the Mobile Communication Group involved in the field of system level simulators for LTE-A. He is actively involved with research on different projects offering wireless communication solutions for industrial environments. He has experience with wireless technologies, such as IEEE 802.11, IEEE 802.15.4, LoRa, LoRaWAN, and SigFox. His current research interests include Internet of Things, Industry 4.0, low power wide area networks, and wireless sensor networks.

Ingrid Moerman received the degree in electrical engineering and Ph.D. degree from Ghent University, Ghent, Belgium, in 1987 and 1992, respectively,

She became a part-time Professor with Ghent University, in 2000. She is a Staff Member with the Internet and Data Laboratory Research Group (IDLab), a core research group of imec with research activities embedded at Ghent University and the University of Antwerp, Antwerp, Belgium. She is leading a research team of about 30 members at IDLab, Ghent University. She is coordinating the research activities on mobile and wireless networking. She has a longstanding experience in running and coordinating national and EU research funded projects. At the European level, she is in particular very active in the future networks research area, where she has coordinated and is coordinating several FP7/H2020 projects (CREW, WiSHFUL, eWINE, and ORCA) and participating in other projects (Fed4FIRE, FORGE, FLEX, and Flex5Gware). She has authored or co-authored over 700 publications in international journals or conference proceedings. Her current research interests include Internet of Things, low power wide area networks, high-density wireless access networks, collaborative and cooperative networks, intelligent cognitive radio networks, real-time software defined radio, flexible hardware/software architectures for radio/network control and management, and experimentally supported research.

Jeroen Hoebeke received the master's degree in engineering (computer science) and Ph.D. degree in adaptive *ad hoc* routing and virtual private ad hoc networks from Ghent University, Ghent, Belgium, in 2002 and 2007, respectively.

Since 2002, he has been with the Internet and Data Laboratory (IDLab) Research Group, Department of Information Technology, Ghent University—imec, where he has been an Assistant Professor since 2014, leading research on mobile and wireless networks, Internet of Things (IoT) communication solutions, and embedded communication stacks. His expertise has been applied in a variety of IoT domains, such as logistics, Industry 4.0, building automation, healthcare, and animal monitoring. He has authored or co-authored over 90 publications in international journals or conference proceedings. His current research interests include solutions for realizing the IoT covering wireless connectivity (802.11, 802.15.4, BLE, LoRa, and 802.11ah), standard-based solutions (IETF CoAP, IPv6, IPSO, and OMA LWM2M), distributed intelligence, robust wireless communication, deployment and self-organization of smart objects, application enablers, wireless diagnosis, realizing the IoT covering wireless connectivity, standard-based solutions, distributed intelligence, robust wireless communication, deployment and self-organization of smart objects, application enablers, and wireless diagnosis.