Open Innovation Platform for IoT-Big data in Sub-Sahara Africa

Grant Agreement Nº 687607

Report D2.2

Deliverable Title: Low-latency and low-energy MAC for IoT connectivity

---

Responsible Editor:	UPPA
Contributors:	UPPA
Document Reference:	D2.2 – Low-latency and low-energy MAC for IoT connectivity
Distribution:	Public
Version:	1.0
Date:	March 9th, 2018

CONTRIBUTORS TABLE

DOCUMENT SECTION	AUTHOR(S)	REVIEWER(S)
Section 1	C. PHAM, UPPA	A. RAHIM
Section 2	C. PHAM, UPPA E. MUHAMAD, UPPA	M. SHEIKHALISHAHI
Section 3	C. PHAM, UPPA E. MUHAMAD, UPPA	T. TEIXEIRA
Section 4	M. DIOP, UPPA C. PHAM, UPPA	C. DUPONT

DOCUMENT REVISION HISTORY

Version	Date	Changes
V1.0	MARCH 9TH, 2018	PUBLIC RELEASE
v0.5	MARCH 2ND, 2018	VERSION FOR INTERNAL APPROVAL
v0.4	MARCH 2ND, 2018	INTEGRATION OF REVIEWS
v0.3	FEB 12TH, 2018	FIRST RELEASE FOR REVIEW
v0.2	FEB 8TH, 2018	SECOND DRAFT
v0.1	JAN 15TH, 2018	FIRST DRAFT

EXECUTIVE SUMMARY

This deliverable 2.2 entitled « Low-latency and low-energy MAC for IoT connectivity » covers the achievements between month 12 and month 24 of T2.2 and T2.3. It will be structured as follows:

- 2. LORA ACTIVITY SHARING to add quality and service and indirectly reduce transmission latencies in duty cycle environments.
- 3. OPTIMIZED CARRIER SENSE to reduce packet collisions thus transmission cost and latencies
- 4. 2-HOP LORA CONNECTIVITY to improve coverage, reduce both packet losses and transmission cost

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. BRIEF REVIEW OF WAZIUP AND WP2

1.1. WAZIUP

WAZIUP is a collaborative research project using innovative technological research applications on IoT and related big data management and advanced data analytic techniques. It has the support of multiple African stakeholders and public bodies with the aim of defining a new innovation space to advance the African Rural Economy. The potential of IoT and Big Data, in Sub-Saharan Africa, can be realized only if the cost is reasonable as most of the rural population in the Africa is at the poverty level. This is the main challenge that WAZIUP will address. In addition, WAZIUP is creating developer communities and innovation hubs to train, adapt, validate and disseminate results. WAZIUP's technical partners will develop methodologies, tools, software libraries and "recipes" for building low-cost IoT and data analysis platforms. Tightly involving end-users communities in the loop, namely rural African communities of selected pilots, will ensure quick appropriation and easy customization by third parties. Furthermore, WAZIUP organizes frequent training and Hackathon sessions in the sub-Saharan African region. WAZIUP will tackle the challenges enlisted below:

a) Challenge 1: Innovative design of the IoT platform for the Rural Ecosystem. Low-cost, generic building blocks for maximum adaptation to end-applications in the context of the rural economy in developing countries.
b) Challenge 2: Network Management. Facilitate IoT communication and network deployment. Lower cost solutions compared to state of the art technology: privilege price and single hop dedicated communication networks, energy autonomous, with low maintenance costs and long lasting operations.
c) Challenge 3: Long distance. Dynamic management of long range connectivity (e.g., cope with network & service fluctuations), provide devices identification, abstraction/virtualization of devices, communication and network resources optimization.
d) Challenge 4: Big-data. Exploit the potential of big-data applications in the specific rural context.

1.2. WP2

The objectives of WP2 are to co-design, adapt and deploy sensing systems with low-power and low-cost long-range (LR) communication and networking infrastructure. WP2 will set a test-bed in UGB for validation and performance evaluation of use cases in various rural areas. The outcomes from WP2 with corresponding tasks are listed below:

a) T2.1 Design and adaptation of sensing systems considering societal and environmental threat - design, adapt and develop sensing systems for IoT nodes that will be deployed and tested in the use cases. Produce open-source plug-&-sense platforms for fast, easy deployment & customization to use cases. Methodology and tools for low-cost design.

b) T2.2 Design and integration of heterogeneous IoT networking - integrate the hardware components and develop the software building blocks for generic usage of LR radio technologies. Develop and evaluate IoT activity scheduling algorithms that ensure fairness and efficiency in heterogeneous radio technologies communications. Data management for seamlessly supporting intermittent networking.

c) T2.3 Low-latency and low-energy MAC protocols - address the issue of achieving low-latency and low-energy communications with the combination of LR and short/medium range radios. Propose and evaluate IoT node activity scheduling based on the timing requirement of the end application.

d) T2.4 Open IoT test-bed setup and benchmark - deploy the IoT test-bed featuring LR IoT device. Provide the validation infrastructure for internal sub-tasks (T2.1, T2.2 and T2.3) and the demonstrator infrastructure for the use cases.

e) T2.5 Training materials and tools for developer community - address the training and dissemination part of WP2. Developed sensing systems and software will be presented and explained to end-developers and end-users. Organize dedicated Living Labs, Tech Events and Hackathon to boost innovations around the IoT technologies and the WAZIUP developed solutions.

1.3. Deliverable D2.2

This deliverable 2.2 entitled « Low-latency and low-energy MAC for IoT connectivity » covers the achievements between month 12 and month 24 of T2.2 and T2.3. It is mainly oriented towards research activities to enhance flexibility and performance of LoRa networks. All the outcomes have been integrated into the WAZIUP IoT communication library developped in WP2 and previously presented in D2.1. D2.2 will be structured as follows:

- 2. LORA ACTIVITY SHARING to add quality and service and indirectly reduce transmission latencies in duty cycle environments.
- 3. OPTIMIZED CARRIER SENSE to reduce packet collisions thus transmission cost and latencies
- 4. 2-HOP LORA CONNECTIVITY to improve coverage, reduce both packet losses and transmission cost

2. LoRa Activity Sharing (LAS)

2.1. Introduction

Wireless Sensor Networks (WSN) and so-called Internet of Things (IoT) devices are typically envisioned as the fundamental building blocks in a large variety of smart digital ecosystems: smart cities, smart agriculture, logistics&transportation to name a few. However, the deployment of such devices in a large scale is still held back by technical challenges such as short communication distances. Using the traditional telco mobile communication infrastructure is still very expensive (e.g. GSM/GPRS, 3G/4G) and not energy efficient for autonomous devices that must run on battery for months. During the last decade, low-power but short-range radio such as IEEE 802.15.4 radio have been considered by the WSN community with multi-hop routing to overcome the limited transmission range. While such short-range communications can eventually be realized on smart cities infrastructures where high node density with powering facility can be achieved, it can hardly be generalized for the majority of surveillance applications that need to be deployed in isolated or rural environments. Future 5G standards do have the IoT orientation but these technologies and standards are not ready yet while the demand is already high. Therefore, recent modulation techniques are developed to achieve longer transmission distances without relay nodes to reduce the deployment complexity independently from the mobile telecom industry. Rapidly adopted by many Machine-to-Machine (M2M) and IoT actors the concept of Low-Power Wide Area Networks (LPWAN), operating at much lower bandwidth, is gaining incredible interest. Some low-power long-range technologies such as Sigfox are still operator-based. However, other technologies such as LoRa proposed by Semtech radio manufacturer can be privately used and deployed following the recently proposed LoRaWAN specifications [1] or on a completely ad-hoc manner. The long-range solution has the following advantages over traditional short-range technologies:

1) avoid relying on operator-based communications; no subscription fees;
2) remove the complexity and cost of deploying/maintaining a multi-hop infrastructure;
3) can offer out-of-the-box connectivity facilities.

Figure 1 shows a typical long-range 1-hop connectivity scenario to a long-range base station (LR-BS) which it is the single interface to Internet servers, e.g. cloud computing services. Most of long-range technologies can achieve 20km or higher range in line-of-sight (LOS) condition and about 2km in non-LOS condition such as dense urban areas where the radio frequency (RF) signal has to travel through several buildings [8,12]. Most of long-range technologies can achieve 20km or higher range in line-of-sight (LOS) condition and about 2km in non-LOS condition such as dense urban areas where the RF signal has to travel through several buildings [8,12].

The flexibility of long-range transmission comes at the cost of stricter legal regulations. For instance, in Europe, electromagnetic transmissions in the EU 863-870MHz Industrial-Scientific-Medical (ISM) band used by Semtech's LoRa technology falls into the Short Range Devices (SRD) category. The ETSI EN300-220-1 document [3] specifies various requirements for SRD devices in Europe, especially those on radio activity. Basically, a transmitter is constrained to 1% duty-cycle (i.e. 36s/hour) in the general case. Note that this duty-cycle

limitation approach is also adopted in China in the 779-787MHz ISM Band. US regulations in the 902-928MHz ISM Band do not specify duty-cycle but rather a maximum transmission time per packet with frequency hopping that is not directly addressed by our work presented here.

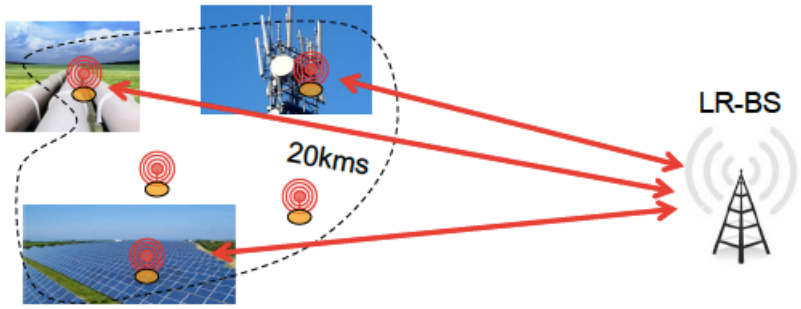

Figure 1 – Extreme long-range application

While this activity time may be large enough for most of devices and for most of scenarios, it still remains the issue of what to do when a device still needs to transmit critical information and has exhausted its allowed activity time in the current period of time, even when all possible optimization mechanisms have been applied (e.g. data aggregation, adaptive data rate, etc.). As these devices are mostly considered to be deployed for surveillance applications and infrastructures, we believe this issue is highly important to address in order to somehow provide quality of service guarantees when deploying these applications: these devices should not simply stop transmitting nor violate deliberately the regulation. In addition, there is a growing need for going beyond simple physical measures such as temperature or luminosity, with the possibility to provide visual information for a number of surveillance applications [20,6,2,13]. Based on our experience in image sensors [16,17] we developed a long-range LoRa version of the image sensor capable of transmitting an 128x128 image at medium quality in about 15s to 20s using an optimized image encoding mechanism. The advantage of long-range radio is clearly to remove the multi-hop constraints and provide out-of-the-box visual capabilities for a various number of surveillance applications. The work presented in this part of the document is greatly motivated by the limitations of such imaging systems when using long-range radio in the unlicensed radio spectrum. However, our proposed mechanism can be applied to a larger variety of applications, including of course traditional telemetry applications.

In this chapter, we address the case of deploying a pool of remote devices, managed by a single organization under duty-cycle regulations. We proposed to overcome the tight 36s/hour radio activity of a device by considering all the sensor's individual activity time in a shared/global manner. In a real deployment scenario, sensors deployed to monitor different areas or infrastructures will unlikely be activated by critical events occurring at the same time at different places. It is therefore expected that most sensors will not use all their allowed radio activity time most of the time. The approach we proposed will allow a device that needs to go ``exceptionally" beyond the activity time limitation to borrow some from other devices in order to provide better surveillance service guarantees. A global view of the total activity time allowed per 1-hour cycle will be maintained so that each device knows the remaining activity time in the current cycle. We would like to emphasize on the fact that the proposed mechanism is not intended to be used on a regular basis, i.e. where a device is commissioned to always report data at a rate that makes it consuming more than the

allowed duty-cycle limitation, but only to provide a ``last-chance" solution for providing better surveillance service guarantees while globally satisfying duty-cycle regulations.

This chapter of the document is organized as follows. Section 2.2 reviews the LoRa long-range technology and motivates quality of service (QoS) research for long-range sensor deployed for surveillance applications. Going beyond simple sensors, we illustrate the need of QoS with our long-range image sensor. In Section 2.3, we present the proposed Long-range Activity Sharing (LAS) mechanism that considers all the sensor node's individual activity time in a shared/global manner. We also discuss some issues such as the cost of adding LAS services, how to support sleep period and the usage of Inter Frame Spacing (IFS) to reduce collisions. Our LAS implementation and experiments are described in Section 2.4. In particular, we show how LAS services can be easily added to existing devices. Section 2.4 will also present real-world deployment design choices such as how Channel Activity Detection (CAD) for increased reliability is implemented, how device synchronization for network startup and supporting sleep period is realized and how new devices can be dynamically added during network operation.

2.2. QoS for long-range sensors in unlicensed radio spectrum

2.2.1. Long-range Semtech LoRa technology

Semtech's LoRa technology uses an enhanced chirp spread spectrum modulation technique to provide much higher receiver sensitivity (thus longer range) while keeping the transmission power low [22,4]. Mainly working in the sub-GHz unlicensed radio bands, such transmissions are then constrained in many countries by maximum radio duty-cycle [3]. This duty-cycle can be 0.1%, 1% or 10% depending on the frequency band or transmission power. This duty cycle limit applies to the total transmission time, even if the transmitter can change to another channel. For LoRa radios, the relevant measure is the time-on-air (`ToA`) which depends on the 3 main LoRa parameters: `BW`, `CR` and `SF`. `BW` is the physical bandwidth for RF modulation (e.g. 125kHz) and larger signal bandwidth allows for higher effective data rate, thus reducing transmission time at the expense of reduced sensitivity improvement. `CR` is the coding rate to perform forward error detection and correction. Such error coding incurs a transmission overhead and the lower the coding rate, the higher the coding rate overhead ratio, e.g. with `CR=4/5` the overhead ratio is 1.25 which is the minimum value. Finally, `SF`, the spreading factor that can be set from 6 to 12. The lower the `SF`, the higher the data rate transmission, but the lower the immunity to interference thus the smaller is the range.

Figure 2 shows for various combinations of `BW` and `SF` settings the ToA when the payload size is varied (so-called LoRa mode defined in [11]). We use a communication library with a 5-byte header overhead : a total payload of 55 bytes means that 50 bytes are available for the application. We use the formula given by Semtech in [21] to compute the `ToA` for all these so-called LoRa modes. Mode 4 to 6 provide quite interesting tradeoffs for longer range, higher data rate and immunity to interference. Mode 1 provides the longest range.

With appropriate hardware support, a LoRa gateway can listen to multiple channels at different spreading factors. This is how a LoRaWAN network can implement the so-called

Adaptive Data Rate (ADR) with end-devices using different spreading factor values depending on their distance to the gateway. By using a smaller spreading factor, the ToA is reduced therefore a larger amount of data can be sent within the 36s of allowed transmission time. However, as the interest of LoRa technology resides mainly in the longer range, the benefit of ADR is very limited when a large number of end-devices are placed far from the gateway.

Listen Before Talk (LBT) along with Adaptive Frequency Agility (AFA) can be used to go beyond the 1% duty-cycle limit but then additional restrictions are introduced: the Tx on-time for a single transmission cannot exceed 1s. If this 1s limit is respected, then the transmitter is allowed to use a given channel for a maximum Tx on-time of 100s over a period of 1 hour for any 200kHz bandwidth. The advantage is that using AFA to change from one channel to another, longer accumulated transmission time is possible. One major drawback is that if a sensor works following the LBT+AFA scheme to benefit from the 100s ToA, then in Figure 2 all ToA greater than 1s cannot be used. If we look at mode 4, then the maximum payload that can be used is 114B. However, if we want maximum range by using mode 1 for instance, we can see that most payload sizes have ToA greater than 1s! Note that this issue can be the main reason to prefer the 1% duty-cycle approach to the LBT+AFA approach.

				time on air in second for payload size of					
LoRa mode	BW	CR	SF	5 bytes	55 bytes	105 bytes	155 Bytes	205 Bytes	255 Bytes
1	125	4/5	12	0.95846	2.59686	4.23526	5.87366	7.51206	9.15046
2	250	4/5	12	0.47923	1.21651	1.87187	2.52723	3.26451	3.91987
3	125	4/5	10	0.28058	0.69018	1.09978	1.50938	1.91898	2.32858
4	500	4/5	12	0.23962	0.60826	0.93594	1.26362	1.63226	1.95994
5	250	4/5	10	0.14029	0.34509	0.54989	0.75469	0.95949	1.16429
6	500	4/5	11	0.11981	0.30413	0.50893	0.69325	0.87757	1.06189
7	250	4/5	9	0.07014	0.18278	0.29542	0.40806	0.5207	0.63334
8	500	4/5	9	0.03507	0.09139	0.14771	0.20403	0.26035	0.31667
9	500	4/5	8	0.01754	0.05082	0.08154	0.11482	0.14554	0.17882
10	500	4/5	7	0.00877	0.02797	0.04589	0.06381	0.08301	0.10093

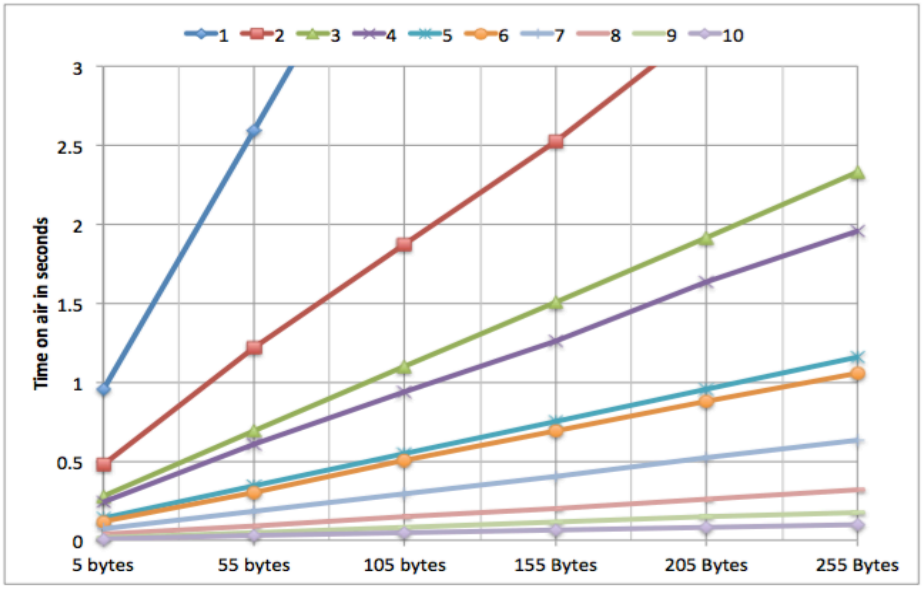

Figure 2 – Time on air for various LoRa modes as payload size is varied

2.2.2. The need of QoS for advanced surveillance applications

Working with duty-cycle restrictions obviously poses quality of service issues when important data still need to be reported when the allowed transmission time is exhausted. As indicated previously, even a traditional low data rate telemetry system (for reporting temperature for instance) needs some service guarantees. With growing interest and needs for going beyond simple physical measures such as temperature, the possibility to provide visual information with small images will impose larger amount of bytes to be transmitted. In this case the duty-cycle limit can be reached very rapidly. Our developed image sensor works with raw 128x128 8-bbp gray scale image which is encoded using an encoding method optimized for low-resource platforms [10], see Figure 3(right). This encoding scheme features the following 2 key points: (`i`) image compression must be carried out by independent block coding in order to ensure that data packets correctly received at the sink are always decodable and, (`ii`) de-correlation of neighboring blocks must be performed prior to packet transmission by appropriate interleaving methods in order to ensure that error concealment algorithms can be efficiently processed on the received data. For these reasons, the encoded bit stream is particularly tolerant to packet losses which is highly important for very low-bandwidth and high-latency technologies such as long-range radios. Additionally, a tuning parameter, called Quality Factor (`Q`), provides a compression ratio/energy consumption trade-off that can further be used to optimize transmission time.

Figure 3(left) shows the original raw 128x128 image taken with the image sensor and encoded with various quality factors: `Q=90` (high quality), `Q=50` (medium quality) and `Q=10` (low quality).

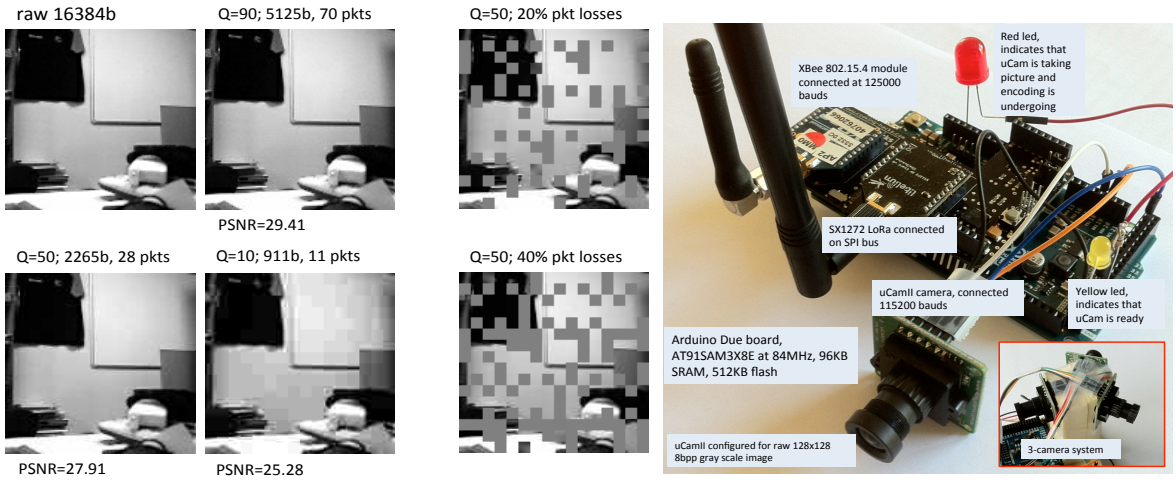

Figure 3 – 128x128 image taken by the image sensor, various quality factor

The total size of the compressed image, the compression ratio (in bracket) the number of generated packets and the `PSNR` compared to the original image are shown. The number of packets depends on the image maximum segment size (`IMSS`) allowed per packet. For Figure 3(left) we set it to 90B. The produced packet size will slightly vary according to the packetization process. The 90-byte limit was initially defined for 802.15.4 radio where the maximum MAC payload size is 100B. Semtech's LoRa radio can handle up to 255B payload in variable packet size mode. As the cost of packet sending is usually reduced using larger

packets it is more beneficial to increase the IMSS close to the radio limit. When removing various header overheads, 240B remain available. Figure 4 summarizes for various quality factors the image encode time (column E), the number of produced packets (column N), the encoded image size with the compression ratio (column S) and the packetization time (column P) for IMSS=90 and IMSS=2400. All these measures are taken without transmission of packets.

		MSS=90			MSS=240		
	E	N1	S1	P1	N2	S1	P2
Quality Factor Q	encode time	number of packets	size in bytes (compression ratio)	packetization time	number of packets	size in bytes (compression ratio)	packetization time
100	387	159	10549 (1.55)	629	52	10617 (1.54)	465
90	515	75	5476 (2.99)	204	25	5552 (2.95)	174
80	513	54	4031 (4.06)	135	19	4091 (4)	121
70	520	42	3272 (5)	106	15	3289 (4.98)	101
60	512	35	2800 (5.85)	94	13	2832 (5.78)	89
50	504	30	2447 (6.69)	83	11	2458 (6.66)	82
40	519	28	2196 (7.46)	78	10	2196 (7.46)	76
30	519	23	1817 (9.01)	68	8	1824 (8.98)	69
20	518	17	1428 (11.47)	60	7	1425 (11.49)	62
10	519	11	920 (17.8)	50	4	922 (17.77)	55
5	519	7	555 (29.52)	44	3	553 (29.62)	48

Figure 4 – Image encoding/packetization performance. All times are in ms

With a low quality factor such as 10, our test image is compressed to 922 bytes sent in 4 packets: first 3 packets carry 255 bytes and the last packet has 217 bytes (including header bytes). Although the image encoding process is highly optimized and worked well for multi-hop IEEE 802.15.4 radio [14], it is still a much higher volume of data when using very low bandwidth LPWAN such as LoRa than for traditional telemetry systems. For instance, the consumed transmission time in the longest range mode (mode 1) is about 35s. With the strict duty-cycle limit, only 1 image could be sent per hour!

Lowering the image quality factor can adapt the image sensor transmission strategy to long-range constraints. For instance, the image sensor can use a much lower quality factor (5 or even smaller) to reduce the encoded image size, thus reducing overall ToA of the image, when remaining activity time is small. However, this strategy has its limit and may allow for an additional image at a very low quality to be transmitted but does not really solve the quality of service issue.

2.3. Providing QoS by Activity Time Sharing

We propose to provide QoS for long-range radio working in unlicensed spectrum with an additional mechanism which considers the activity time of all deployed long-range devices in a shared manner. An organization deploying a pool of *n* long-range devices can use up to a Global Activity Time of $G_{AT} = n \times DC_{AT}$ per hour, where $DC_{AT} = 36000ms$ is the regulated maximum local activity time (1% duty-cycle per hour). Note that we express time in ms to avoid complex floating point variable coding. Then, the basic idea is to allow each long-range device to use up to G_{AT} and have knowledge of its evolution over the 1-hour period.

There are basically 2 approaches for devices to update their knowledge of a global resource such as our proposed G_{AT}: a decentralized or a centralized approach. In a decentralized approach, devices can listen for exchanged messages and compute their corresponding ToA to decrease G_{AT} accordingly. This behavior is basically similar to many so-called passive listening approaches that have been proposed in some MAC control mechanisms such as [9,24]. However, since devices usually go to sleep mode most of the time to save energy, listening to radio activity can not be done without increasing dramatically the energy consumption or without dedicated hardware support such as low power listening [19]. In addition, the hidden terminal problem where 2 devices may not be visible to each other makes a decentralized approach very costly in terms of message exchanges for maintaining the system's consistency. Therefore, in practice, a decentralized approach is not tractable as it will be discussed in more details later on. We propose a centralized approach where the LR-BS updates G_{AT} on reception of packets from remote devices and will broadcast new values for G_{AT} at appropriate moment as it will be explained later on. The centralized approach is quite well adapted to the way long-range infrastructures are working, with the LR-BS acting as the single point of interface from remote devices to Internet servers.

2.3.1. Synopsis

We present in the following paragraphs the outline of our proposed centralized radio activity sharing approach. The core mechanism is not complex as the main objective is to provide for all devices willing to share their activity time the Global Activity Time (G_{AT}) of the system and their remaining amount of activity time (local or remote). There is therefore a simple initialization phase where devices would register with the LR-BS and receive back the value of G_{AT}. This initialization phase is described below.

Action 1 – *Initialization*

a) All deployed long-range devices D_i sharing their activity time initially register (REG packet) with the LR-BS by indicating their default local Remaining Activity Time l^i_{RAT0}. The LR-BS stores all l^i_{RAT0} in a table (the last l^i_{RAT0} value is also saved), computes G_{AT} and broadcasts (INIT packet) both n (the number of devices) and G_{AT}, see Figure 5 (left). For sake of simplicity we assume that all devices start at t_0 and that they share 100% of their local activity time. It is possible to handle different startup time and a fraction of local activity time (by indicating a $l^i_{RAT0} < DC_{AT}$ in the registration message). Note that this step is performed periodically every hour.

b) On reception of n and G_{AT} from the INIT message each device D_i can consider an initial (and locally managed) $G^i_{AT} = l^i_{RAT0} + \sum_{j=1,j\neq i}^{n} l^i_{RAT0}$, as shown in Figure 5 (right)(a). D_i also sets its local Remaining Activity Time, l^i_{RAT0} (the green bar), to l^i_{RAT} and both its local Total Activity Time, l^i_{TAT}, and its remote Activity Time Usage, r^i_{ATU}, to 0. It is possible in the `INIT` message to limit the G_{AT} ratio allowed for usage to $\alpha \times G^i_{AT}$. We assume here that α `= 100%`.

After initialization, each device starts its sensing task and will periodically report to the LR-BS by sending data packets as illustrated previously in Figure 1. As indicated in action *1b*, the initial G_{AT} from the LR-BS is locally managed by each device: every uplink data packet sent by

device D_i actually decreases G_{AT}^{j} of all devices $D_j \neq i$. For device D_i, G_{AT}^{i} is unchanged while its local Remaining Activity Time (l_{RAT}^{i}, the green bar) is decreased by the ToA of the packet, see Figure 5(right)(b). The objective of our proposed mechanism is therefore to make each device D_i aware of the evolution of their local G_{AT}^{i} as global activity time is decreased by uplink data packets. The interesting case, which motivates our work, is when a device D_i "exceptionally" needs to go beyond its allowed activity time to report important information, i.e. $l_{RAT}^{i} < ToA(pkt)$. This is when our mechanism will allow such device to borrow remote activity time (the red region) as illustrated in Figure 5(right)(c). The extra consumed activity time should be taken from the l_{RAT}^{i} of the other devices $D_j \neq i$ and new values for both G_{AT}^{j} and l_{RAT}^{i} should reflect the new activity time amount.

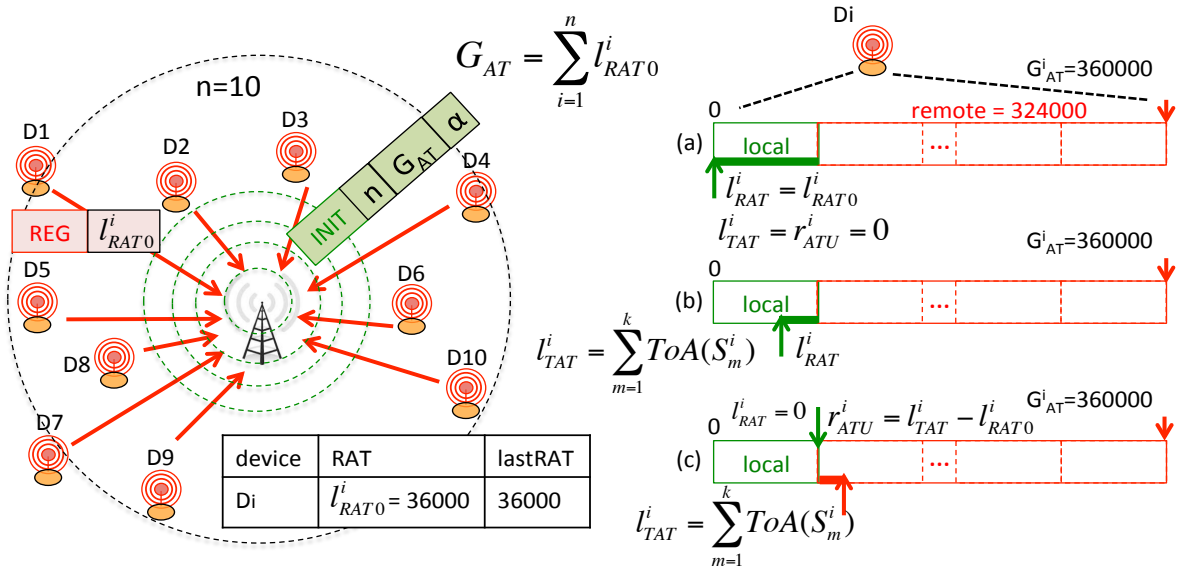

Figure 5 – Left: initialization. Right: device's local and remote activity time

The consistency of the system is mainly managed by the LR-BS which uses downlink update messages to indicate corrections to apply to both G_{AT}^{j} and l_{RAT}^{j}. While the core approach for updating radio activity time for each sensor is not new and is quite simple in its objectives, being similar in some way to synchronized MAC [25] or cluster-based routing [5], doing so in the context of duty-cycled long-range radio is challenging: the consumption of activity time by the additional control messages needs to be kept to a minimum and still providing an acceptable level of coordination. The rest of this Section will describe in more details our proposed activity time sharing mechanism, explaining how the cost of control messages can be kept minimum, how the entire system can run even under sleep-active behavior and how transmission of control packets can be made more robust. Section 4, will then present an implementation and will further describe the additional issues that have been added to support real-world deployment scenarios.

2.3.2. Detailed description

We describe in more detail our proposed centralized radio activity sharing approach. For clarity purposes, the list of the parameters and their explanation is shown below.

Description	Notation	Remark
Number of devices participating in the LAS pool	n	a device may decide to not share its activity time
Device i	D_i	indicates device i
Maximum duty-cycle activity time	DC_{AT}	36000ms, i.e. 1% duty cycle
Size of packet k sent by D_i	S_k^i	expressed in bytes
Time on Air for S bytes	$ToA(S)$	computed by both devices and LR-BS
Global Activity Time	G_{AT}	activity time of the entire system, computed and broadcasted by LR-BS
Maximum G_{AT} ratio usage	α	default is set to 1 by LR-BS
Locally managed G_{AT}	G_{AT}^i	initially set by D_i to G_{AT}
Default local Remaining Activity Time	l_{RAT0}^i	default is set by D_i to DC_{AT}. Is sent to LR-BS on registration. LR-BS will maintain its own copy in a table for each D_i
Local Remaining Activity Time	l_{RAT}^i	initially set by D_i to l_{RAT0}^i
Remote Activity Time Usage	r_{ATU}^i	initially set by D_i to 0
Local Total Activity Time	l_{TAT}^i	initially set by D_i to 0
Cumulated Activity Time	AT^i	computed by LR-BS for D_i
Previous value of l_{RAT0}^i	$lastl_{RAT0}^i$	maintained by LR-BS in a table for D_i
A list of device's id	$E\{D_k\}$	defined and broadcasted by LR-BS

Action 2 – Device D_i sending the kth DATA packet of size S_k^i

a) D_i computes $ToA(S_k^i)$.
b) If $l_{TAT}^i + ToA(S_k^i) > \alpha \times G_{AT}^i$ then ABORT.
c) D_i updates $l_{TAT}^i = l_{TAT}^i + ToA(S_k^i)$ and $l_{RAT}^i = l_{RAT}^i - ToA(S_k^i)$, see Figure 4 (right)(b).
d) If $l_{TAT}^i > l_{RAT0}^i$ then D_i sets $l_{RAT}^i = 0$ and $r_{ATU}^i = l_{TAT}^i - l_{RAT0}^i$ (the red bar), see Figure 5 (right)(c).
e) If $r_{ATU}^i > 0$ puts r_{ATU}^i in data packet and sets the Remote Activity Time Usage (RATU) flag; otherwise, puts l_{RAT}^i in data packet.

Action 3 – LR-BS receives the kth DATA(l_{RAT}^i) packet from D_i of size S_k^i

a) LR-BS computes $ToA(S_k^i)$ and updates for device D_i $l_{RAT0}^i = l_{RAT0}^i - ToA(S_k^i)$.
   1) If received $l_{RAT}^i < l_{RAT0}^i$ updates for device D_i $l_{RAT0}^i =$ received l_{RAT}^i
b) If received $l_{RAT}^i > l_{RAT0}^i$ (probably due to a reset of the end-device) then use the SET flag for UPDT message. When last packet or timeout from D_i computes $AT^i = l_{RAT0}^i - last\ l_{RAT0}^i$.
   1) If $l_{RAT0}^i > 0$, broadcasts an UPDT message indicating $|AT^i|$ and D_i's id. If SET flag then replaces $|AT^i|$ by l_{RAT0}^i.
   2) If $l_{RAT0}^i < 0$, then determines how many devices, n_d, should take over the extra activity time consumed by device D_i and broadcasts an UPDT message with a Remote Activity Time Usage (RATU) flag indicating $|AT^i|$, D_i, $|l_{RAT0}^i|$, n_d and a list of device's id. If $last\ l_{RAT0}^i < 0$ then $|AT^i|$ is replicated in the $|l_{RAT0}^i|$ field as D_i had already consumed all its local activity time. For the selected devices j, the LR-BS updates their l_{RAT0}^j (stored in the table) accordingly, $l_{RAT0}^j = l_{RAT0}^j - |l_{RAT0}^i|/n_d$, and sets $last\ l_{RAT0}^j = l_{RAT0}^j$.
   3) If an UPDT message has been sent, saves the current value of l_{RAT0}^i into $last\ l_{RAT0}^i$.

Action 4 – Device D_j receiving an UPDT($|AT^i|, i$) from LR-BS

a) If $j \neq i$ then D_j updates $G_{AT}^j = G_{AT}^j - |AT^i|$.

b) If $j = i$ and UPDT w/SET flag then updates $l_{RAT}^j = |AT^i|$ and device is not participating in the shared pool until next cycle: $G_{AT}^j = l_{RAT0}^j$ and $l_{TAT}^j = l_{RAT0}^j - |AT^i|$.

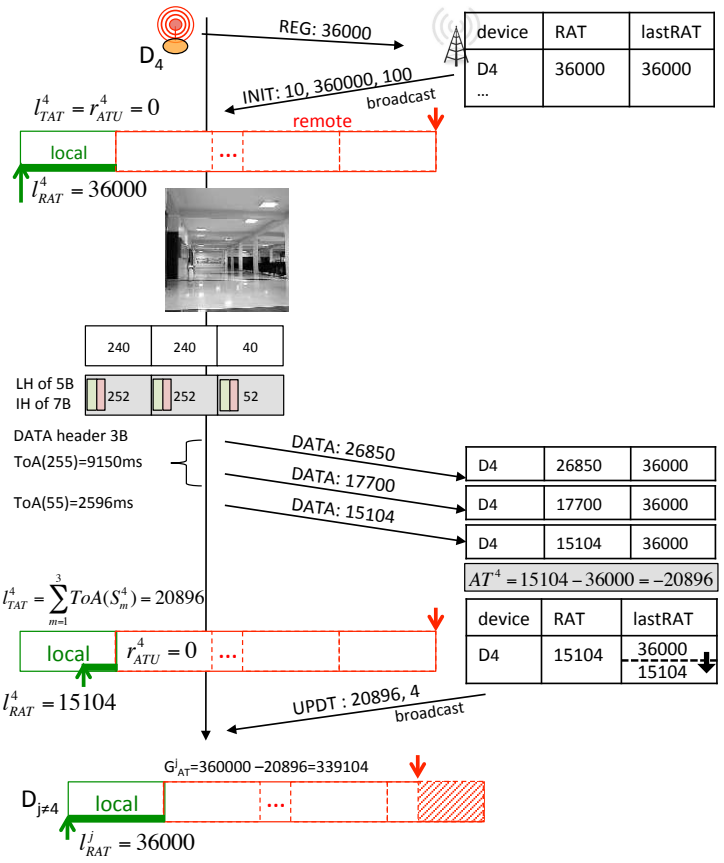

Figure 6 – Initialization and sending of DATA packets

We explain actions 2, 3 and 4 by taking as use case a visual surveillance application with image devices capable of transmitting still images either on an on-demand basis or with an automated event detection mechanism. We therefore illustrate in Figure 6 a whole sequence with an image transmission from device D_4. We assume that LoRa mode 1 is used for maximum range, which somehow represents the worst case in term of packet time-on air. After the initialization phase, D_4 knows the value of $G_{AT} = 360000ms$ and the number of devices, $n = 10$. We also assume that the LR-BS indicates that devices could use 100% of the G_{AT}^i, i.e. $\alpha = 100$.Upon image change detection, D_4 encodes the image and we assume that 3 packets are produced: the encoded image size is taken on purpose to make the computation of ToA easy with Figure 2. IMSS is set to 240B and the last packet is 40B long. Using our own image platform specifications [16,17], adding the protocol header and the image header results in increasing the size of each packet by 12B. As the DATA packet needs 3 additional bytes, the ToA for LoRa mode 1 is computed on 255B except for the last packet where the final size is 55B. At the source device D_4, each packet transmission of size S_k^4 will remove $ToA(S_k^4)$ from l_{RAT}^4, see action 2c. At some point, it may happen that $l_{RAT}^4 < 0$

which means that the 1% duty-cycle allowed for D_4 has all been used. This is checked in action 2d and D_4 will start using remote activity time by recording its remote Activity Time Usage r^4_{ATU}. This case will be presented later on.

In our proposed approach, the LR-BS keeps track of consumed activity time when receiving a packet from a device D_i. In Figure 6, we can see at the LR-BS side that each data packet triggers action 3a at the LR-BS. However, it may happen that a packet sent by D_i is not received by the LR-BS, for some reasons, while the activity time of this unsuccessful transmission should be counted. To make the system more robust to packet losses, the image data packet header includes for device D_i the value of either l^i_{RAT} or r^i_{ATU}, depending on a Remote Activity Time Usage (RATU) flag in the packet header (we will detail in the next section the packet format for our proposed activity time sharing protocol). Actions 2c and 2d are therefore extended by action 2e that indicates l^i_{RAT} or r^i_{ATU} in the data packet header. Then action 3a at the LR-BS includes action 3a.1 which performs a comparison between the l^i_{RAT} indicated in the packet and the l^i_{RAT0} stored by the LS-BR. A different value means that there have been some packet losses and the value indicated in the packet will be used to update the value stored by the LR-BS. In Figure 6, we illustrate how each DATA packet from D_4 carries the value of l^4_{RAT}. Note that if the received $l^i_{RAT} > l^i_{RAT0}$ it probably means that the end-device has somehow reset. We use the SET flag in UPDT message to indicate to the end-device that it should correct its l^i_{RAT} value. This is performed by subsequent actions 3b and 4b. In this particular case, the end-device is removed from the current running pool until next cycle. It can still send messages but will not be able to use remote activity time.

After the last image packet from D_4, or timeout for D_4 the LR-BS computes AT^4 with both l^i_{RAT0} and $lastl^i_{RAT0}$ stored in the table, $AT^i = l^i_{RAT0} - lastl^i_{RAT0} = 15104 - 36000 = -20896$ with action 3b. AT^4 represents the amount to consumed activity time for the image transmission. Here, we still have $l^4_{RAT0} > 0$ therefore action 3b.1 is performed by broadcasting an UPDT message indicating $|AT^4| = 20896$ and D_4's id. With this first image transmission, D_4 only use its local activity time, without having to borrow extra activity time from other devices therefore the UPDT message is sent without the RATU flag with action 3b.1. All devices $j \neq 4$ receiving the UPDT message from the LR-BS should update their local value of G^j_{AT} which should now be decreased by the activity time consumed by D_4, see action 4a. At the bottom of Figure 6, all $D_j \neq 4$ have a new value of $G^j_{AT} = G^j_{AT} - |AT^4|$. As can be seen in Figure 6, for any device D_j the green arrow (l^j_{RAT}) and the red arrow ($local\ G^j_{AT}$) delimit the amount of total allowed activity time for that device when $l^j_{RAT} > 0$.

Figure 7 continues the scenario started in Figure 6: D_4 has only $l^4_{RAT} = 15104ms$ of local activity time left and the LR-BS has updated both l^4_{RAT0} and $lastl^4_{RAT0}$ in its table, see action 3b.3. In Figure 7 needs to send another image. After sending the first packet, $l^4_{RAT} = 5954ms$. The transmission of the second packet is only made possible by our proposed activity time sharing mechanism: $l^4_{TAT} = 39196$ becomes greater than $l^4_{RAT0} = 36000$ therefore action 2d keeps track of the remote activity time usage and in action 2e, the DATA packet is sent with the RATU flag which indicates that the value carried by the DATA packet is now the remote Activity Time Usage, i.e. $r^4_{ATU} = 3196$. The transmission of the third image and last image packets continues to use remote activity time up to a total of $r^4_{ATU} = 14942$. For D_4, its local Total Activity Time l^4_{TAT} is now $50942 = 36000 + 14942$.

At the LR-BS side l^4_{RAT0} is updated at each DATA packet reception as previously with action 3a. If we assume no packet losses, $|l^4_{RAT0}|$ is finally equal to r^4_{ATU} indicated in the last DATA packet. If this is not the case, then the LR-BS can set $l^4_{RAT0} = -r^4_{ATU}$ (because of the RATU flag). Then, again as previously, after the last image packet from D_4, or timeout for D_4 the LR-BS computes AT^4 with both l^4_{RAT0} and $lastl^4_{RAT0}$ stored in the table, $AT^4 = l^4_{RAT0} - lastl^4_{RAT0} = -14942 - 15104 = -30046$ with action 3b. The difference now compared to the previous image transmission is that $l^4_{RAT0} < 0$ so action 3b.2 is executed instead of action 3b.1.

The main idea behind the activity sharing proposition is that although the regulation stipulates an 1% duty-cycle limit, a device can accumulate more than the $30046ms/h$ of activity provided that other devices, deployed by the same organization, will remove some activity time from their normal 1% duty-cycle amount. Therefore, in action 3b.2, the LR-BS determines how many devices, n_d, should take over the extra activity time consumed by device D_4 and broadcast an UPDT message with the Remote Activity Time Usage (RATU) flag indicating $|AT^4|$, D_4, $|l^4_{RAT0}|$, n_d and a list of device's id, $E = \{D_k\}$. In the example of Figure 7, the LR-BS will decide to assign to devices D_5 and D_6, $E = \{D_5, D_6\}$, the role of supporting the extra activity time consumed by D_4, i.e. $|l^4_{RAT0}| = 14942$. Therefore, the UPDT message with the RATU flag starts with the value of $|AT^4| = 30046ms$ followed by D_4's id, $n_d = 2$, $|l^4_{RAT0}| = 14942$ (from the table) and finally D_5 and D_6 ids. Note that in action 3b.2, the LR-BS also updates both l^j_{RAT0} and $lastl^j_{RAT0}$ of selected devices j. After sending the UPDT message, the LR-BS executes action 3b.3 and now $lastl^4_{RAT0} = -14942$.

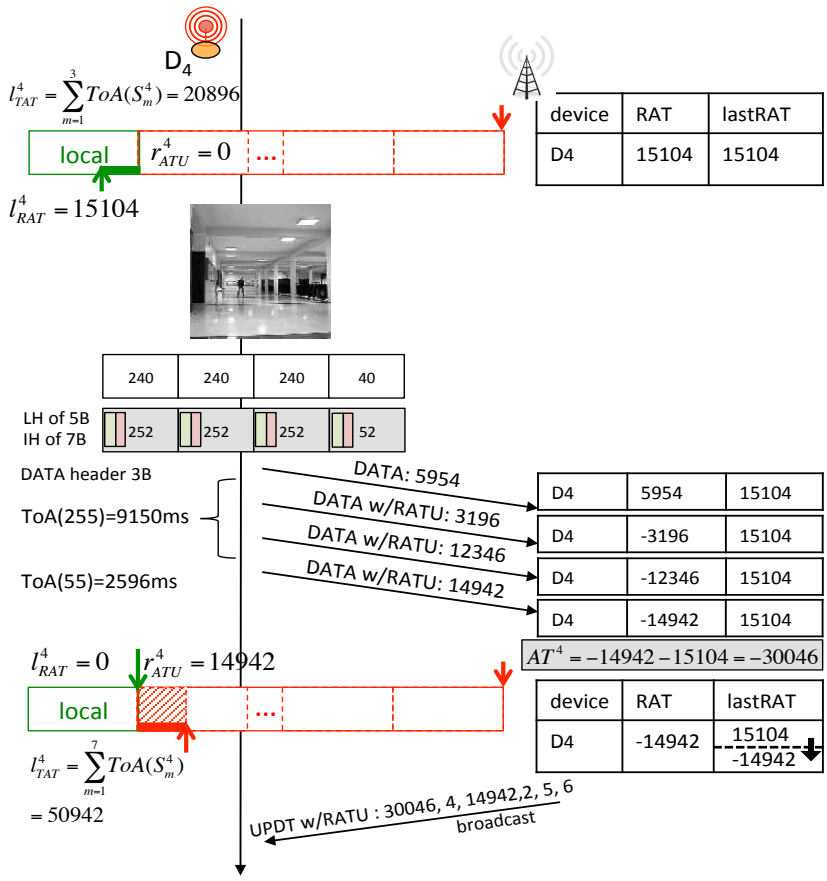

Figure 7 – Consumption of remote activity time

In general, the LR-BS should avoid removing for each device j an activity time greater than l^j_{RAT}. In this paper, we are not evaluating nor proposing a particular selection mechanism since our implementation currently distributes the consumed activity time over all the other devices (use of the ``All Device" flag) to distribute the remote activity time usage over the largest number of devices, thus reducing the amount of activity time penalization for a single device. However if it is desirable to have priority or exclusion mechanisms some particular devices can be selected and in this case it makes sense to select those with the highest l^j_{RAT0}, similar to how devices are chosen according to their energy level in energy-based routing protocols [5] for instance.

The next step is to define the behavior of a device receiving an UPDT message with the RATU flag which indicates that a device has consumed extra activity time and that therefore its local value of G_{AT} should be updated accordingly. This is performed by the following action 5.

Action 5 – Device $\boldsymbol{D_j}$ receiving an UPDT($\left|\boldsymbol{AT^i}\right|, \boldsymbol{i}, \left|\boldsymbol{l^4_{RAT0}}\right|, \boldsymbol{n_d}, \boldsymbol{E} = \{\boldsymbol{D_k}\}$) w/RATU from LR-BS

5a) if $D_j \in E\{D_k\}$, takes the advertised $\left|l^i_{RAT0}\right|$ and updates $l^j_{TAT} = l^j_{TAT} + \frac{\left|l^i_{RAT0}\right|}{n_d}, l^j_{RAT} = l^j_{RAT} - \frac{\left|l^i_{RAT0}\right|}{n_d}$ and $G^j_{AT} = G^j_{AT} - \left|AT^i\right| + \left|l^i_{RAT0}\right|$ because all D_j in list of devices contribute to $\left|l^i_{RAT0}\right|$. See Figure 8.

5b) if $D_{j\neq i} \notin E\{D_k\}$, updates $G^j_{AT} = G^j_{AT} - \left|AT^i\right|$ because it has to remove what has been consumed by D_i. See Figure 8.

5c) if $D_{j=i}$ and UPDT w/SET flag then updates $l^j_{RAT} = 0, G^j_{AT} = l^j_{RAT0}, r^j_{ATU} = l^i_{RAT0}\ and\ l^j_{TAT} = l^j_{RAT0} + \left|l^i_{RAT0}\right|$. The device will not be able to send more messages as $l^j_{TAT} > G^j_{AT}$ so action 2b would prevent any further remote activity time usage.

To explain action 5 with our example, Figure 8 illustrates in more details this important action of our proposition.

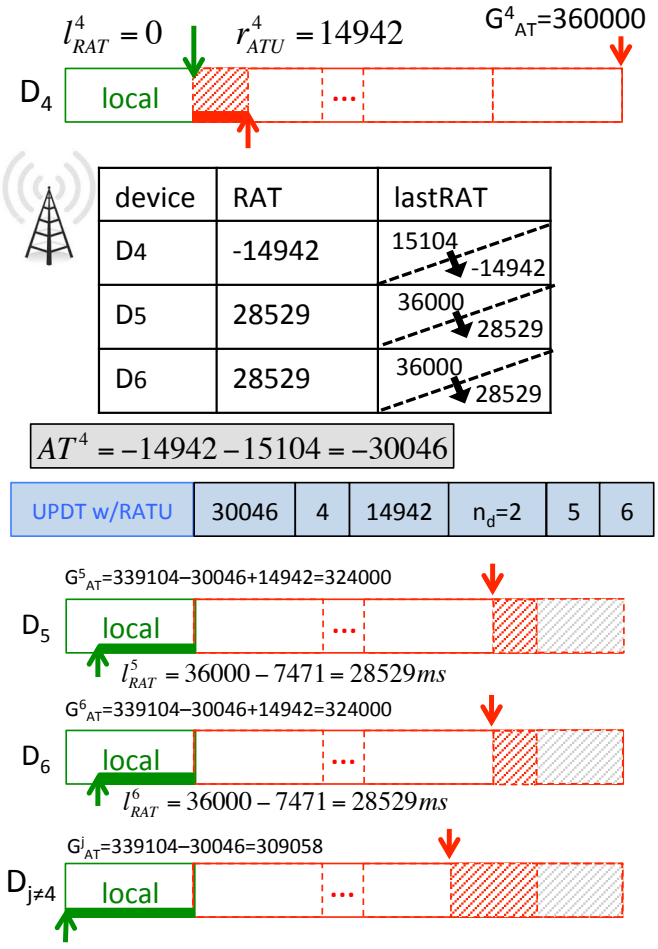

Figure 8 – Update of device's local Remaining Activity Time

Figure 8 starts with the broadcast of the UDPT message by the LR-BS, which is the end of the scenario depicted by Figure 7. D_5 and D_6 that recognized themselves in the list of device's id execute action 5a: they will each remove $\frac{14942}{n_d} = \frac{14942}{2} = 7471$ from l^5_{RAT} *and* l^6_{RAT} respectively. If we assume that both devices did not send any message, then $l^5_{RAT} = l^6_{RAT} = 28529ms$. They then update their local value of $G^{5,6}_{AT}$ by removing AT^4, but adding $|l^4_{RAT0}|$ because both of them already contributed previously to $|l^i_{RAT0}|$. Therefore, at the end, they both have their $G^{5,6}_{AT}$ decreased by D_4's whole duty-cycle, i.e. $360000 - 36000 = 324000ms$. Note that the gray area for a device j is what has been removed from G^j_{AT} at the previous step.

For a device $D_{j\neq 4}$ and NOT in the device list $E\{D_k\}$, it has to remove from its local value of G^j_{AT} the totality of what has been consumed to have a consistent view for G_{AT}, see action 5b. As can be seen in Figure 8, for any device D_j the green arrow (l^j_{RAT}) and the red arrow ($local\ G^j_{AT}$) again delimit the amount of total allowed activity time for that device when $l^j_{RAT} > 0$.

Now, we can continue the scenario by sending 2 more packets, each packet needing 9150ms of activity time. This part of the scenario is interesting as we start with device D_4 that has already consumed extra activity time, see Figure 9.

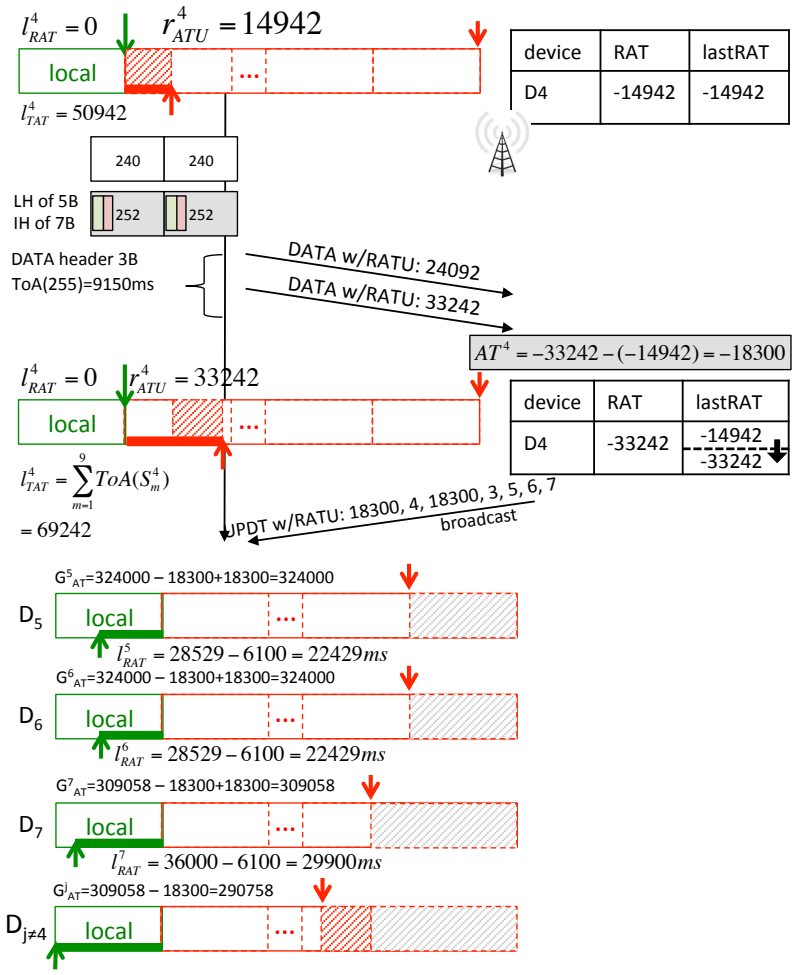

Figure 9 – Continuing using remote activity time

By consuming 18300ms of additional remote activity time, D_4 finishes the transmission of the 2 packets with $r^4_{ATU} = 33242$. Once again, under the no packet loss assumption, the LR-BS has $l^4_{RAT0} = -33242$. Like previously, in action 3b.2, the LR-BS determines how many devices, n_d, should take over the extra activity time consumed by device D_4 and broadcast an UPDT message with the Remote Activity Time Usage (RATU) flag indicating $|AT^4|$, D_4, $|l^4_{RAT0}|$, n_d and the list of device's id $E\{D_k\}$. However, this time, we have $last\ l^4_{RAT0} < 0$, meaning that D_4 was already in the "red zone" with all its local activity time already consumed. Therefore, when the LR-BS broadcast the UDPT message, the $|l^4_{RAT0}|$ field is set with $|AT^4|$. In this way, all devices j in the list of selected devices will correctly update their l^j_{RAT} with action 5a while ending with no modification in their G^j_{AT} because no additional local activity time has been consumed apart from what has already been taken from their l^j_{RAT}. In the example of Figure 9, G^j_{AT}, D_6 and D_7 have been selected and each device, by executing action 5a, decreases $l^{5,6,7}_{RAT}$ by $\frac{|AT^4|}{n_d} = \frac{18300}{3} = 6100$ while maintaining the same

value for their $G_{AT}^{5,6,7}$. All other devices $D_{j\neq4}$ NOT in the list simply remove $|AT^4|$ from their G_{AT}^{j} with action 5b.

If we continue on the scenario by assuming that device G_{AT}^{j} now needs to send image packets, then actions 2d and 2e will determine whether the RATU flag should be positioned for the LR-BS to broadcast the correct UDPT w/RATU message. We then see that the most important UPDT messages are those indicating the consumption of remote activity time.

2.3.3. Packet format

We describe in Figure 10 the packet format of our proposed activity time sharing protocol. The first byte, DSP, contains two 4-bit fields for flag indicators and command type value.

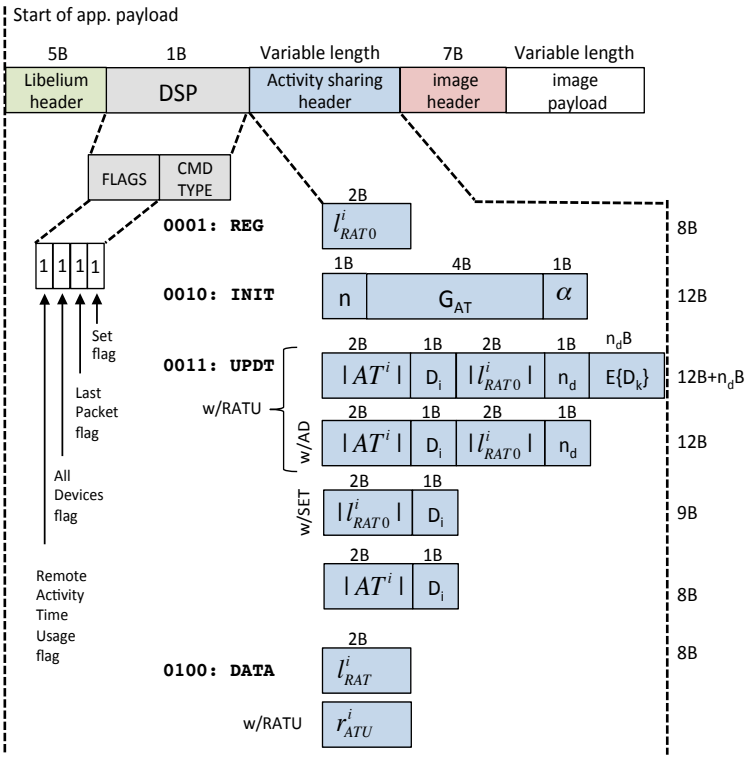

Figure 10 – Packet format

The RATU flag can be set to indicate whether the DATA packet carries an l_{RAT}^{i} or an r_{ATU}^{i} coded in the first 2 bytes after the DSP byte. The LP flag indicates that it is the last packet for the current transaction. This flag can be set at the sender to better determine when the LR-BS can build UPDT messages. However, as the LR-BS also uses a timeout for each device, this flag is not mandatory.

A DATA packet needs 3B plus 5B for the lower level LoRa header, and plus 7B for the image header. Therefore, the total DATA packet size with an IMSS of 240B is 255B which is the limit of the SX1272/76 radio in variable packet size mode. For the UPDT message, if the number of selected devices is greater than 243 (as the UPDT message needs 7B plus 5B for the Libelium header and 1B per device id, only 243 device id can be indicated in a single UPDT message), then another UDPT message can be sent for the remaining selected devices.

However, the LR-BS can select all devices except the one that sent the image by using the All Devices flag (w/AD) to indicate that all devices $j \neq i$ should process the UPDT message with either action 4 or 5.

2.3.4. Cost of the additional protocol message exchanges

2.3.4.1. Reg & INIT messages

The REG message should be sent every hour by an end-device sharing its activity time. As indicated in Figure 10, the total size of a REG message is 8B. If we assume LoRa mode 1 for achieving maximum range, the ToA is 1122ms which should be removed from the l^4_{RAT0} indicated in the REG message. However, if sufficient activity time remains, it is possible for a device to send the REG message prior to the beginning of a new 1h-cycle in order to consume the activity time of the current cycle to start a new cycle with the full activity time, i.e. 36000ms. Therefore, in most cases, the overhead of the REG message can be assumed to be zero otherwise it represents about 3% of the allowed transmission time. The LR-BS is also constrained by the activity time regulation. The size of the INIT message is 12B which results in a ToA of 1286ms. However, the INIT message is also sent once every hour and the same anticipation mechanism than for the REG message can be used here.

Note that a LoRaWAN network also requests that end-devices at least register with the gateway with a `join request` message with the gateway sending an acknowledgment. Therefore, compared to a LoRaWAN network, our approach has similar level of overhead at the device side. If our proposed mechanism has to be integrated into a LoRaWAN network, the `join request` can carry the REG message information.

2.3.4.2. DATA messages

In DATA packets, the additional cost of the activity time sharing mechanism is 3B which can results in a ToA increase of 163ms or zero depending on the rounding effect on the total payload size. As most of computations are performed by the LR-BS when DATA packets are received, the activity time cost introduced by the activity time sharing mechanism is very low compared to the cost of the other protocol headers, e.g. application and network key, message integrity code, ...

2.3.4.3. UPDT messages

An UPDT w/RATU message can use a large portion of ToA if the number of selected devices in large. For instance, selecting 6 devices gives a size of 18B which results in a ToA of 1449ms. For large networks, using the “All Devices” flag can always limit the UPDT w/RATU to 12B. For regular UPDT(D_i) message, i.e. without RATU flag, the cost of action 3b.1 can be made smaller by only triggering the UPDT message if l^i_{RAT0} becomes close to 0. If l^4_{RAT0} is still large the system can easily run until the next update. A regular UPDT(D_i) has also a cumulative effect, meaning that only the latest UPDT needs to be sent. For that purpose, action 3b.3 only copies l^i_{RAT0} into $lastl^i_{RAT0}$ if an UPDT message has been sent, otherwise, the next update will compute the correct $AT^i = l^i_{RAT0} - lastl^i_{RAT0}$. We will describe in section \ref{building_updt} how the implementation can further reduce the number of UPDT messages by using the cumulative behavior.

2.3.4.4. Increasing the LR-BS activity time

With the "All Devices" selection method, each UDPT message is 12B long, with a ToA of 1122ms each when assuming LoRa mode 1 for longest range. As the INIT message has already consumed 1122ms of the LR-BS activity time, it can send up to 31 UDPT messages per hour. While this number of UPDT messages is not limiting with a small pool of a few devices, it may become an issue with much larger number of devices. In this case, the LR-BS can borrow activity time of end-devices in a very straightforward and consistent way: the $|AT^i|$ and $|l^i_{RAT0}|$ field of an UDPT w/RATU message can both be increased by the ToA of the UPDT message itself. To do so, the ToA of all messages sent by the LR-BS will be removed from the LR-BS's own l^{BS}_{RAT} counter (initially set to 36000ms and reset every 1-hour cycle). Prior to send an UPDT w/RATU message $u(|AT^i|, D_i, |l^i_{RAT0}|, n_d, D_j)$ (or the w/AD variant without the list of devices D_j) in action 3b.2, if $l^{BS}_{RAT} - ToA(u) < 0$ then the LR-BS updates $\left|AT^i\right| = \left|AT^i\right| + ToA(u)$ and $|l^i_{RAT0}|$=$\left|l^i_{RAT0}\right| + ToA(u)$. On reception of the UPDT w/RATU message, the selected devices will automatically share the extra remote activity time advertised by the LR-BS in action 5a. For all devices not in the list of devices, action 5b remains exactly the same because it does not matter whether $\left|AT^i\right|$ has been consumed only by device D_i or by devices D_j in the device list to support the ToA of the UPDT w/RATU message.

2.3.4.5. Comparison with a descentralized approach

We provide here some comparisons with a theoretical decentralized approach under the assumption that devices can always be active to listen for other transmitted packets. In a first step, we also take the assumption that all devices can hear each other, with no hidden terminal issues.

Under these assumptions, a DATA packet sent by D_i can be heard by the other $D_{j\neq i}$ that can decrease accordingly their local G^j_{AT}. If the DATA is sent with the RATU flag, then all $D_{j\neq i}$ can decrease their l^j_{RAT} by an equal amount thanks to the knowledge of n, the number of devices indicated in the INIT message. The only restriction here is that the remote activity time usage is distributed on all $D_{j\neq i}$ (the "All Device" method) as it is not tractable to perform a distributed "election" procedure to determine which devices should take in charge the remote activity time usage. In this case, the decentralized approach does not need any UPDT messages from the LR-BS and radio activity time usage of the LR-BS can be limited to only a message per hour indicating the beginning of a new cycle.

In a second step, the assumption that all devices can hear each other is not true anymore and we have the well-known hidden terminal issues. In that case, it is necessary to use the gateway to reach all devices just as in the RTS/CTS mechanism of IEEE 802.11 except if the network is partitioned in a number of sectors (even a quadrant division cannot guarantee that 2 devices in the same quadrant can hear each other). If the LR-BS is used then we actually have a centralized approach comparable to our proposed approach and we can see that under realistic operational conditions it is difficult to implement a decentralized approach.

2.3.5. Supporting sleep period

The normal operation mode of an image end-device sensor is to put the radio in sleep mode until a transmission is required (when an image needs to be sent for instance). This is typically the behavior of a so-called LPWAN Class B device. We will not address high-level device activity scheduling in this paper (in [18] we proposed a criticality-based image node activity scheduling to link the image detection rate to the application's level of criticality and the device redundancy level). With end-devices sleeping most of the time, a synchronized radio wake-up mechanism needs to be realized for devices to receive potential UPDT messages. However, as mentioned above, as a regular UPDT(D_i) message has a cumulative effect, the mechanism inherently provides some facility to support sleep periods. For UPDT w/RATU messages, as the $|AT^i|and\ |l^i_{RAT0}|$ only refer to the last image transmission, they do not have a cumulative effect if the list of devices differs. Therefore, all UDPT w/RATU need to be received and processed in order. For the synchronized radio wake-up mechanism we propose that the LR-BS broadcast periodically (e.g. every 5 or 10 minutes for instance) all the UPDT messages that have been generated and queued for transmission, if any. It is possible to perform aggregation of UDPT messages to reduce the cost of the protocol headers. As a consequence, end-devices need to wake up periodically to look for UDPT messages if any. When receiving UPDT messages, each device applies them sequentially. We will describe in the implementation section how this is practically done.

2.3.6. Reliability issues and IFS for packet transmission

Even if we are not using LBT+AFA to avoid the 1% duty-cycle, we propose to use LBT to perform a clear channel assessment (CCA) similar to what is done in a carrier sense mechanism for improving reliability. LBT will be used in conjunction of a priority mechanism similar to the inter-frame spacing (IFS) mechanism of IEEE 802.11: Distributed IFS (DIFS) and Short IFS (IFS) where SIFS < DIFS. Both will be expressed in symbol period. We however propose a simplified back-off mechanism without exponential increase nor frozing & restarting the back-off timer as in WiFi networks. The LBT mechanism is based on the Channel Activity Detection (CAD) feature offered by the Semtech's LoRa chip.

Prior to send a DATA packet, an end-device should see a free channel for at least a DIFS (we will refer to this case as a $DIFS_{CAD}$). If it is the case the packet is transmitted otherwise the device waits for a random number of DIFS without performing CAD. At the end of the waiting period, the device will try again to have a $DIFS_{CAD}$. This process is repeated until the packet can be transmitted. However, to decrease the probability of concurrent image transmission, while the first DATA packet of an image uses the DIFS, all following packets of the same image will use SIFS. In addition to reduce packet collisions and activity time wastage, having sequential image transmission facilitates actions 3b and 4 in order to have a consistent view for the various usage of remote activity time.

All control messages sent by the LR-BS use the SIFS: INIT and UPDT messages will therefore have higher priority. For the specific case of REG messages that start every cycle, we propose to wait for a random number of SIFS before checking whether the channel is free for at least an SIFS.

All waiting period draws a random number between $[1, WAIT_{max}]$. The value for $WAIT_{max}$ could be set differently for data and control packets: $WAIT_{max}^{ctrl}$ and $WAIT_{max}^{data}$. Figure 11 summarizes with an example the proposed SIFS and DIFS usage with $WAIT_{max}^{ctrl} = 7$ and $WAIT_{max}^{data} = 4$.

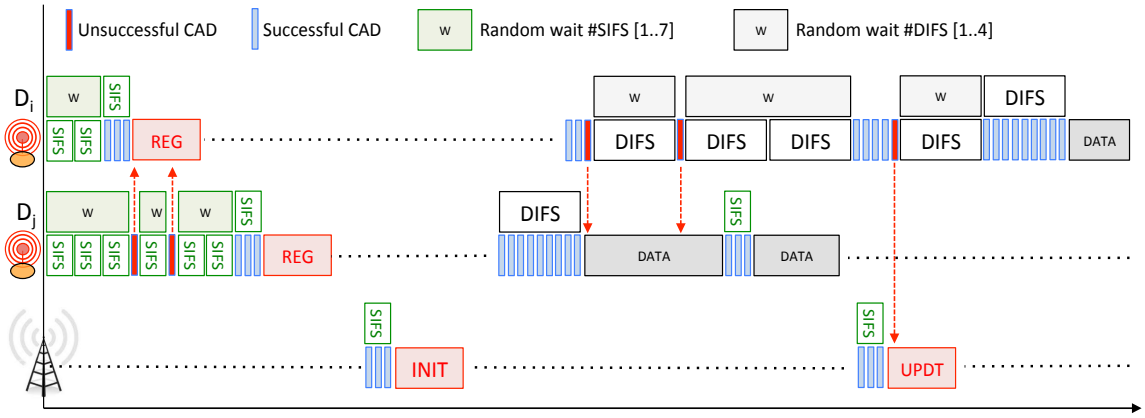

Figure 11 – Example of SIFS and DIFS usage

In the example device D_i draws a random number in $[1, WAIT_{max}^{ctrl} = 7]$ and randomly waits for 2 SIFS while D_j has to wait for 3 SIFS. D_i succeeds to have an SIFS_{CAD} so it can transmit the REG message. D_j tried to have a clear channel twice but could not and have to randomly wait for 1 then 2 SIFS. Finally, the channel becomes free for an SIFS and D_j sends its REG message. After some time, the LR-BS sends the INIT message using an SIFS. Then, D_j is the first to try transmitting 2 image packets. After getting a DIFS_{CAD}, the first packet is transmitted, followed by a second packet after an SIFS_{CAD}. We can see that D_i also tried to get the channel but was unsuccessful 3 times and had to draw 3 random waiting periods in $[1, WAIT_{max}^{data} = 4]$: it has to randomly wait for 1, then 2, then 1 DIFS before getting a DIFS_{CAD}. The first two unsuccessful DIFS_{CAD} were caused by the first image packet from D_j. The third unsuccessful DIFS_{CAD} is due to the UPDT message from the LR-BS that has been sent using SIFS.

2.4. Experiments

We implemented the proposed long-range activity time sharing mechanism on our image sensor platform using the Libelium library for controlling the Semtech SX1272 chip and extending it with Channel Activity Detection (CAD) feature. The LBT and the SIFS/DIFS priority mechanism are therefore implemented using the CAD functionality of the Semtech SX1272 chip. All main described features have been integrated and tested with a pool of 3 image sensors which gives a G_{AT} of 108000ms. As mentioned previously, only the “All Devices” selection method is implemented at the LR-BS. In addition, the LoRa channel and mode is fixed for all devices, including the LR-BS, although the LR-BS can specifically send command to configure an end-device differently. For more advanced MAC layer control mechanisms the activity sharing mechanism can be used on top of a MAC protocol such as the LoRa MAC protocol stack [23] that provides most features of the LoRa Alliance LoRaWAN specification [1], or IBM (with Semtech)'s "LoRa WAN in C" [7] protocol used in their Long-Range Signaling and Control (LRSC) architecture for M2M infrastructures.

Our system can currently handle up to 254 devices as device address 0 is reserved for broadcast and one address (e.g. 0x01) needs to be assigned to the LR-BS. With a maximum of 254 devices per LR-BS, for a single organization, the G_{AT} can be 9144000ms which can easily fit into the 32-bit field reserved in the INIT message.

2.4.1. Defining SIFS and DIFS timer values

The `SIFS` and `DIFS` defined in our proposal, and as in many existing protocols, should be related somehow to the symbol period T_{sym}. For Semtech LoRa T_{sym} depends on the parameters BW and SF as follows: $T_{sym} = \frac{2^{SF}}{BW}$. For instance, Libelium LoRa mode 1 use BW=125kHz and SF=12@ therefore $T_{sym}^{mode-1} = \frac{2^{12}}{125000} = 0.032768$. In our current implementation `SIFS` and `DIFS` however does not depend directly on T_{sym} but on the duration of the CAD mechanism which is a quite complex process as transmissions can be done below the noise floor. In [21], it is reported that the CAD duration is between $1.75T_{sym}$ and $2.25T_{sym}$ depending on the spreading factor. We show the CAD duration expressed in T_{sym} in Figure 12.

In order to simplify the timing of both `SIFS` and `DIFS` we assign to `SIFS` an integer number of CAD duration as shown in Figure 12 in column SIFS/CAD and then define `DIFS=3xSIFS`. We therefore decided to assigned 3 CAD to an `SIFS`. For LoRa mode 1, we will have for instance an indicative value of `SIFS=3xCAD=3x60.948=182.845ms` and `DIFS=3xSIFS=9xCAD=548.536ms`. For the maximum random waiting time, we set $WAIT_{max}^{ctrl} = 7$ and $WAIT_{max}^{data} = 4$. These settings have been used in Figure 11 shown previously, especially for the `SIFS` vs `DIFS` ratio. For instance, the transmission time of an image composed of 8 packets, assuming no contention on the medium, will therefore use 1 `DIFS` for the first packet, then 7 `SIFS`, one for each following packet. The image latency (not the radio activity) will then be increased by 2011ms as shown in the last column of Figure 12.

LoRa mode	Tsym (ms)	CAD duration (Tsym)	CAD duration (ms)	SIFS/CAD	SIFS indicative value (ms)	DIFS=3SIFS indicative value (ms)	increase in latency (8 pkts) (ms)
1	32.768	1.86	60.948	3	182.845	548.536	2011.30
2	16.384	1.86	30.474	3	91.423	274.268	1005.65
3	8.192	1.77	14.500	3	43.500	130.499	478.49
4	8.192	1.86	15.237	3	45.711	137.134	502.82
5	4.096	1.77	7.250	3	21.750	65.249	239.25
6	4.096	1.81	7.414	3	22.241	66.724	244.65
7	2.048	1.75	3.584	3	10.752	32.256	118.27
8	1.024	1.75	1.792	3	5.376	16.128	59.14
9	0.512	1.79	0.916	3	2.749	8.248	30.24
10	0.256	1.92	0.492	3	1.475	4.424	16.22

Figure 12 – SIFS and DIFS values is ms

Our implementation provides a low-level `doCAD(counter)` function that takes an integer number of CAD, i.e. counter, performs sequentially the requested number of CAD and returns 0 if all CAD have been successful (no activity). If one CAD has not been successful, then the function exits with value 1 and, according to our proposed mechanism, the device

has to wait for a random number of SIFS or DIFS. We performed some tests to verify the CAD duration against what is given in [21] and measured the SIFS and DIFS timers which are then implemented as SIFS=doCAD(3) and DIFS=doCAD(9). Figure 13 shows the measures obtained with a 1ms-accuracy clock (the Arduino millis() function). The minimum and the maximum measured values are reported and we can see that the CAD duration is quite consistent with the values shown in Figure 12.

	CAD duration (ms), 1ms granularity			SIFS measured value (ms)		DIFS=3SIFS measured value (ms)		
LoRa mode	min	max	SIFS/CAD	min	max	min	max	DIFS/SIFS
1	60	62	3	182	183	548	549	3.00
2	29	31	3	92	94	277	279	2.97
3	14	16	3	43	44	131	132	3.00
4	15	16	3	46	47	138	139	2.96
5	7	8	3	21	23	65	67	2.91
6	7	9	3	22	24	67	69	2.88
7	3	5	3	11	12	33	34	2.83
8	1	3	3	5	7	17	19	2.71
9	1	1	3	3	5	9	11	2.20
10	0	1	3	2	3	6	7	2.33

8	1	3	6	11	12	34	36	3.00
9	1	1	6	6	7	18	20	2.86
10	0	1	6	3	4	11	13	3.25

Figure 13 – Measured CAD duration, SIFS and DIFS values is ms

The SIFS and DIFS measures are also quite consistent with the theoretical values which indicates that the overhead of the doCAD() function is very low. In the last column, we show the ratio between DIFS and SIFS by dividing the maximum measured value of DIFS by the maximum measured value of SIFS. We can see that for the last 3 lines (LoRa mode 8, 9 and 10 in the upper part of the figure), as the CAD duration is very small the impact of processing overheads (control loops, register reading and testing, …) is higher. Therefore, we can observe that the computed ratios are quite significantly lower than 3 with the real implementation. Since the IFS mechanism described previously requires random waiting for either SIFS or DIFS in case of unsuccessful $SIFS_{CAD}$ or $DIFS_{CAD}$, we want to preserve as much as possible the DIFS=3xSIFS relation: for LoRa mode 8, 9 and 10 we increase the number of CAD per SIFS to 6. Doing so gives more stable ratios between SIFS and DIFS as shown in the bottom part of Figure 13. In addition, by increasing the number of CAD per SIFS when CAD duration is very small, we can have SIFS values that are not too small, e.g. above 3ms. As with the Arduino all function calls are performed from the application layer, small delay values for SIFS may be masked by other processing overheads. We chose the SIFS and DIFS delay value, for a given LoRa mode, to be the maximum value measured in our tests: when testing the channel, SIFS=doCAD(3) and DIFS=doCAD(9) except for LoRa modes 8, 9 and 10 where SIFS=doCAD(6) and DIFS=doCAD(18). When waiting for a random number of SIFS or DIFS we take the maximum value shown in column $SIFS_{max}$ and $DIFS_{max}$, expressed in milliseconds. For instance, in LoRa mode 4, we will wait 47ms for an SIFS and 139ms for a DIFS.

2.4.2. Defining SIFS and DIFS timer values

The LAS features are implemented as a library with a `LAS` base class and 2 derived class, `LASDevice` and `LASBase`, to instantiate an end-device or a LR-BS respectively. The first LAS initialization command is to call the `setSIFS()` function by giving the chosen LoRa mode. Each time that a device changes its transmission mode, `setSIFS()` must be called to adapt the `SIFS/DIFS` duration accordingly. `LASBase` implements all the LR-BS functionalities described previously and interactions with `LASBase` are realized through an `isLASMsg()` function and a `handleLASMsg()` function to handle REG and DATA packets. The basic gateway described previously, has been extended in a straightforward manner with our proposed LAS LR-BS services. For the `LASDevice` class, in addition to `isLASMsg()` and `handleLASMsg()` that are similar to those of `LASBase` to handle INIT & UPDT messages, it mainly provides a `sendData()` function which realizes LAS action 2 as described previously. At the device's initialization, a call to `sendReg()` will send a REG message to the LR-BS. A random waiting delay can be performed at each device by calling the `delayReg()` dedicated function. When receiving the INIT message from LR-BS, the device starts a new cycle. Then, both classes provides a `checkCycle()` function to handle periodic tasks and cycle updates of the LAS services. Therefore, in the main control loop of the host program, a call to `checkCycle()` must be made.

We show below a simple Arduino code for our image device using the LAS services provided by the `LASDevice` class to illustrate how LAS service can be added to an existing project. Lines marked by an '*' are those specifically added for the LAS service.

```
01  #include "SX1272.h"
02  #include <SPI.h>
03* #include "LoRaActivitySharing.h"
04
05  #define LR_BS_ADDR     1
06  #define LORA_ADDR      9
07
08  // provides our own address and the address of the LR-BS
09* LASDevice loraLAS(LORA_ADDR,LR_BS_ADDR);
10
11  int e, s, k;
12  uint8_t loraMode=4;
13  uint8_t buff[300];
14
15  void setup() {
16    Serial.println("Simple sender");
17    // configure the Libelium SX1272 radio module
18    e = sx1272.ON();
19    e = sx1272.setMode(loraMode);
20    e = sx1272.setChannel(CH_10_868);
21    e = sx1272.setNodeAddress(loraAddr);
22    Serial.print("Setting LoRa addr ");
23    Serial.println(loraAddr);
24    // set the correct SIFS duration according to the LoRa mode
25*   loraLAS.setSIFS(loraMode);
26*   loraLAS.ON(LAS_ON_WRESET);
27    // send the REG message
28*   loraLAS.delayReg();
29*   loraLAS.sendReg();
30  }
31
32  void loop() {
33    // called periodically to detect the start of a new cycle
34*   loraLAS.checkCycle();
```

```
35    // Check for message, timeout after 1500ms. We use the Libelium LoRa SX1272
library
36    e = sx1272.receivePacketTimeout(1500);
37
38*   if (!e && loraLAS.isLASMsg(sx1272.packet_received.data)) {
39       Serial.println("Received from LoRa: ");
40       // INIT and UPDT messages will be handled by LAS services.
41       // Provides the source device address and the payload
42*      int v=loraLAS.handleLASMsg(sx1272.packet_received.src,
sx1272.packet_received.data);
43
44       if (v==DSP_DATA) {
45          // normally, devices do not receive data from LS-BS
46          // but we can do whatever is necessary with DATA from LR-BS
47       }
48    }
49
50    // simplified code for image processing
51    bool isLastPacket;
52    if (hasImageChange()) {
53       imageEncode();
54       k=0;
55       while ((s=getNextImagePacket(buff, &isLastPacket))>0) {
56            Serial.print("Send image pkt ");
57            Serial.println(k);
58            // the first image packet needs a DIFS, all following packets use
SIFS
59*           e = loraLAS.sendData(LR_BS_ADDR, buff, s, (k==0)?LAS_FIRST_DATAPKT:
60
((isLastPacket)?LAS_LAST_DATAPKT:LAS_NO_FIRST_DATAPKT));
61
62            if (e==TOA_OVERUSE)
63              Serial.println("-->Packet not sent, TOA_OVERUSE");
64            k++;
65       }
66    }
67  }
```

All the image processing related code have been simplified in the code example to 3 functions: `hasImageChange()`, `imageEncode()` and `getNextImagePacket()` (code lines 52, 53 & 55). We have 3 devices (addresses are 9, 10, 11) that registered with the LR-BS, but only one (the one with address 0x09) is actually sending image data packets in the experiment to make the example simpler to follow. LoRa mode 4 is used (code line 12 & 19). We then show the output provided by the device when using the debug mode of LAS library in Figure 14. Currently, the transmission of REG, INIT & UPDT messages are done without removing their ToA from l_{RAT} by disabling the LAS service for these messages. This is done on purpose here to mainly focus on the image DATA message activity control.

The LR-BS (left) is started first, then the end-devices are started. After initialization and configuration of the radio module (code lines 18-22) , the LAS services are started with `ON(LAS_ON_WRESET)` that also make a reset on LAS statistics (code line 25). Devices send asynchronously the REG message and will wait for the broadcasted INIT messages (code lines 28-29). When receiving the INIT message, they start a new cycle with $G_{AT} = 108000ms$ and store the local time at which their cycle starts. As mentioned previously, we disabled ToA management for REG & INIT messages as can be seen with the `LASBase::disabled` or `LASDevice::disabled` indication when sending these messages. The debug information also shows that CTRL messages are sent with `SIFS` priority. In the experiment, all transmission attempts succeed because devices send the REG message with a random waiting period (code line 27) and then only device 9 is sending packets to the LR-BS. Figure

14 then shows device 9 sending 11 image packets: the first packet uses a DIFS and all following packets use an SIFS (code line 59). We realized some image transmission tests with periodic transmission of data packets (using DIFS) in the background and validated that the DIFS/SIFS mechanism does give higher priority to the current image transmission sequence. Then, we can see how the LAS service takes care of activity time by computing the ToA of packets to be sent and by updating accordingly activity time sharing variables: mainly l_{RAT}^{9}, l_{TAT}^{9} and r_{ATU}^{9}.

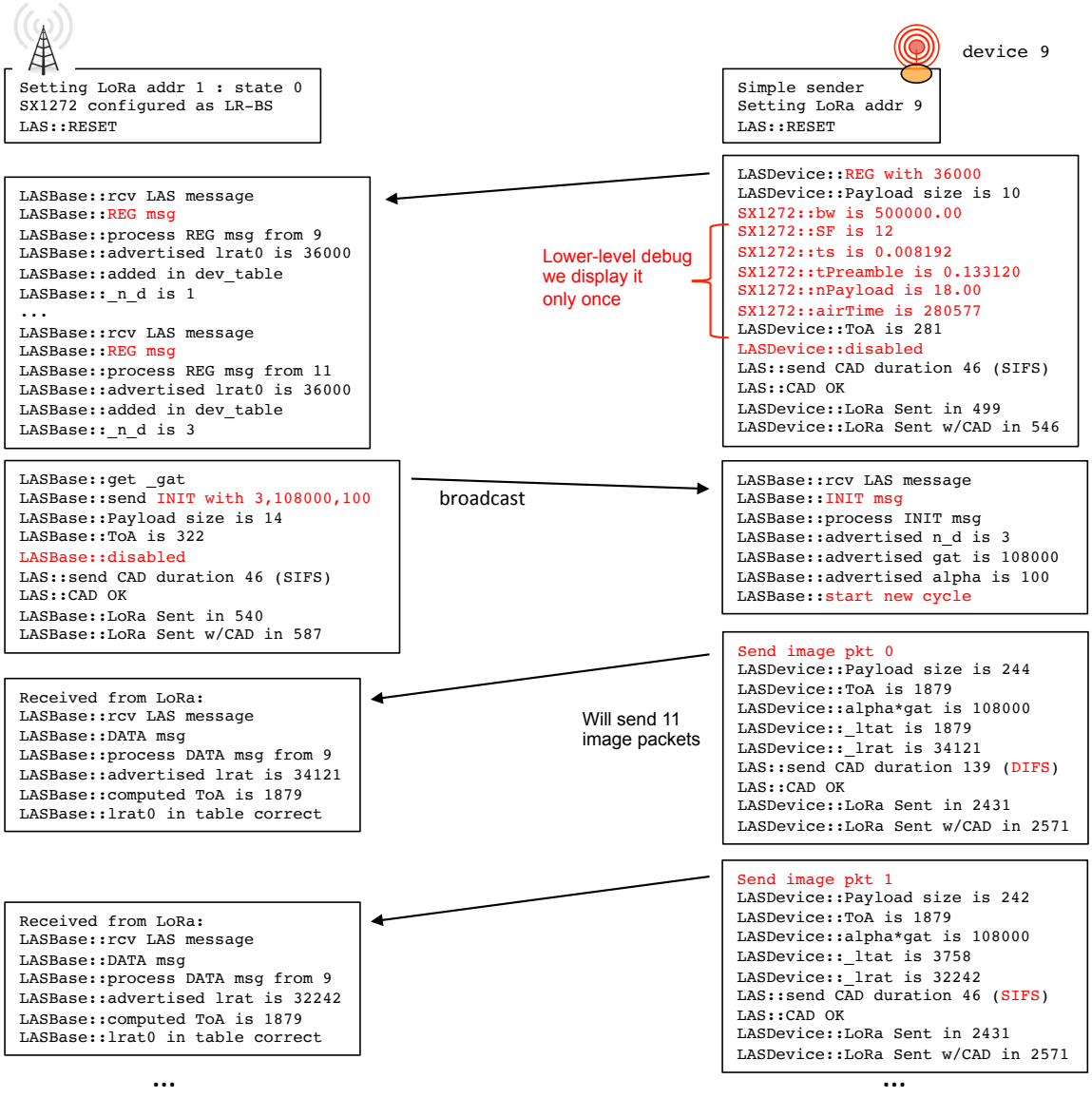

Figure 14 – Example. Step 1: initialization, REG and DATA

Figure 15 illustrates the next step where the LR-BS sends the UPDT message after the last image packet (packet \#10) has been received. After the first image, device 9 has consumed 20053ms of its local activity time: $l_{TAT}^{9} = 20053$ and $l_{RAT}^{i} = 15947$. Then device 9 sends a second image consisting in 10 packets. Here, when sending packet #8, we can see that $l_{TAT}^{9} = 36963$ meaning that device 9 uses remote activity time from other devices. It terminates the image transmission with packet \#9 and shows a $r_{ATU}^{9} = 1367$. This time, the LR-BS sends an UPDT w/RATU w/AD message to inform devices 10 and 11 that they have to update both their l_{RAT}^{j} and G_{AT}^{j}. When device 9 keeps sending many image packets, Figure

15 shows how, at some time, the LAS services prevent device 9 to use additional activity time as l_{TAT}^{9} will become greater than the total allowed activity time for a pool of 3 devices, i.e. $G_{AT} = 108000ms$. Note that even with an incomplete number of image packet, the LR-BS can always decode the received image packets thanks to the robust image encoding scheme described previously in section 2.2.

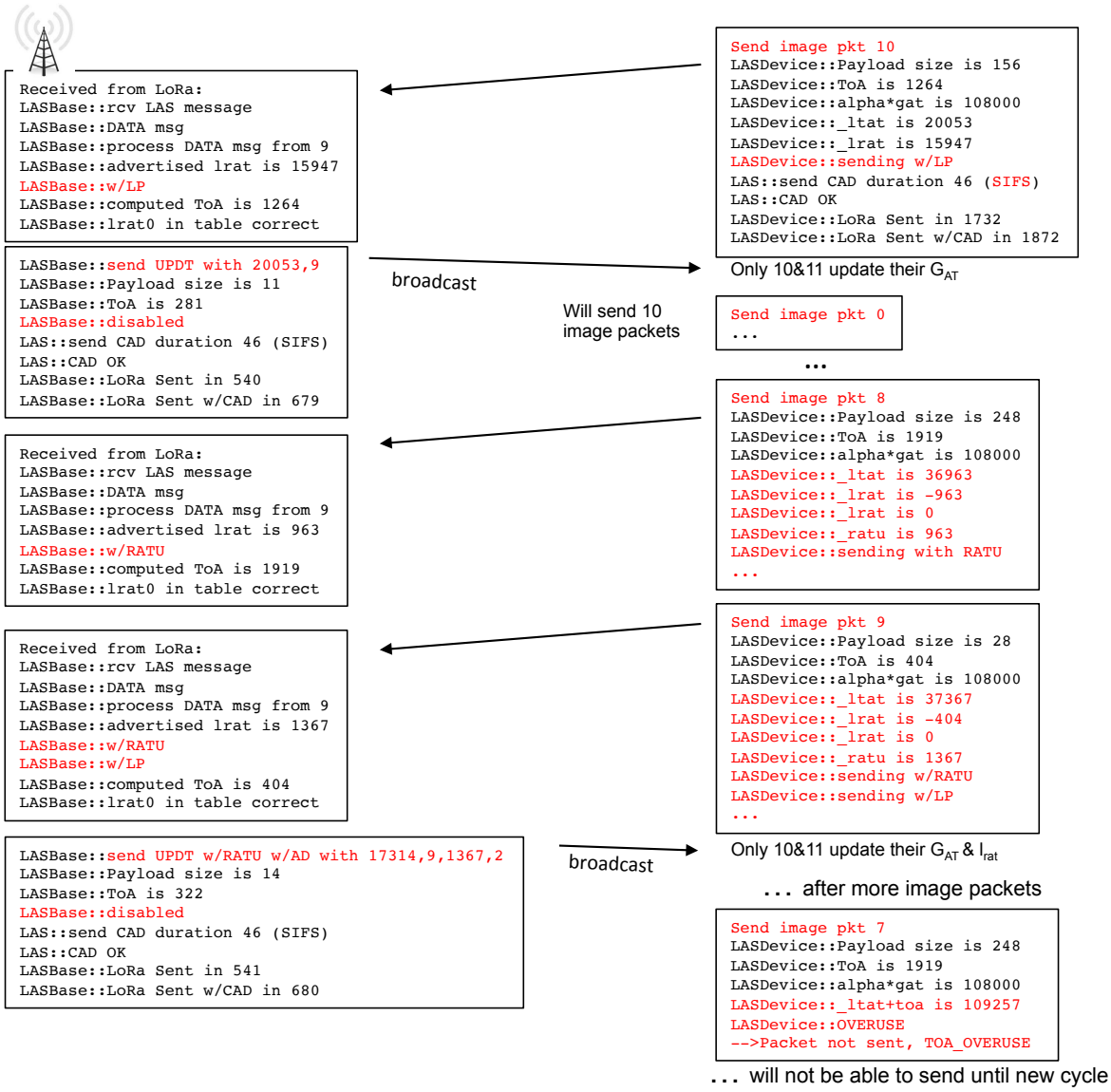

Figure 15 – Example. Step 2: UPDT message and remote activity time usage

Figure 16 shows how statistics can be displayed for both the LR-BS and an end-device by issuing the "/@LASS#" (LAS Statistics) command string. At the LR-BS side, the device table is shown with device j's address, l_{RAT0}^{j} and $lastl_{RAT0}^{i}$. The last column, `needUpdate` indicates whether an UPDT message needs to be sent: 0 means no, and 1 means that an UPDT message is pending. We will explain in the next subsections how the sending of INIT and UPDT messages is currently implemented.

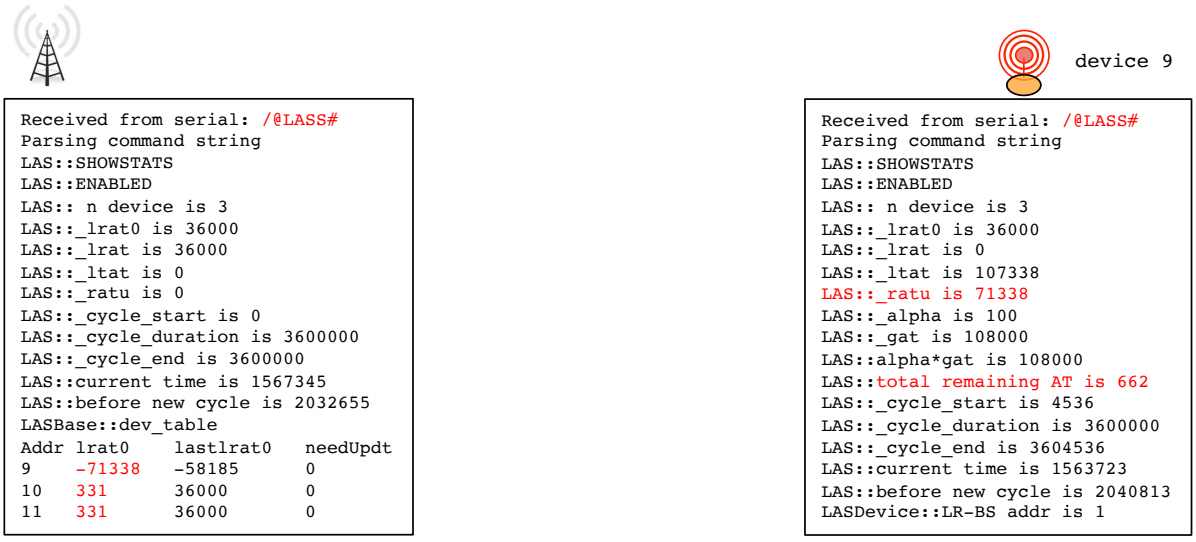

Figure 16 – Getting LAS statistics

2.4.3. Scheduling end-devices for INIT and UPDT messages

In the activity sharing specification, we assumed for simplicity that devices and the LR-BS are started at about the same time, with devices sending a REG message upon startup (see action 1a. In practice it is difficult to ensure that all devices will start in a synchronized manner to send the REG message and to wait for the INIT message in the same time window. As the LR-BS can not afford to send many INIT messages to reach all devices, the system needs to be somehow synchronized to start the LAS services.

Our implementation uses a simple startup mechanism where the LR-BS can issue an INIT message with $n = 0$ indicating a special INIT message, noted $\texttt{INIT}_{\texttt{restart}}$, to initiate a restart procedure where all devices should send their REG message. The G_{AT} field of the INIT is used by the $\texttt{INIT}_{\texttt{restart}}$ to carry an `INIT_DELAY` that indicates when the INIT will be sent. In our implementation, the $\texttt{INIT}_{\texttt{restart}}$ message can be sent by the LR-BS by using the "`/@LASI#`" (LAS Init) command at the LR-BS. In this case, code lines 28-29 can simply be removed. Assuming that the $\texttt{INIT}_{\texttt{restart}}$ is sent at $t_{restart}^{send}$ it will be received at $t_{restart}^{rcv}$ by each active device. Upon reception, each device sends its REG message (using a random waiting time to limit collision) and should listen for the regular INIT message to get knowledge of the total number of devices and of G_{AT}. Actually, after the REG message a device can switch off its radio to only wake up at the appropriate moment to listen for the INIT message which will be sent by the LR-BS at $t_{init}^{send} = t_{restart}^{send} + \text{INIT\_DELAY}$. The difference between $t_{restart}^{send}$ and $t_{restart}^{rcv}$ being very small (less than 1s), a device can expect to receive the INIT at $t_{init}^{rcv} = t_{restart}^{rcv} + \text{INIT\_DELAY}$. In addition, the receive window starts a bit before and ends a bit after the theoretical value for t_{init}^{rcv}, typically a few seconds (`LAS_SECURITY_SYNC`) before and after as illustrated in Figure 17, therefore increasing the probability of correctly receiving the control messages. When the LR-BS does not know the number of devices (first $\texttt{INIT}_{\texttt{restart}}$), it will set `INIT_DELAY = LAS_MAX_INIT_DELAY` otherwise `INIT_DELAY = LAS_INIT_DELAY x n`.

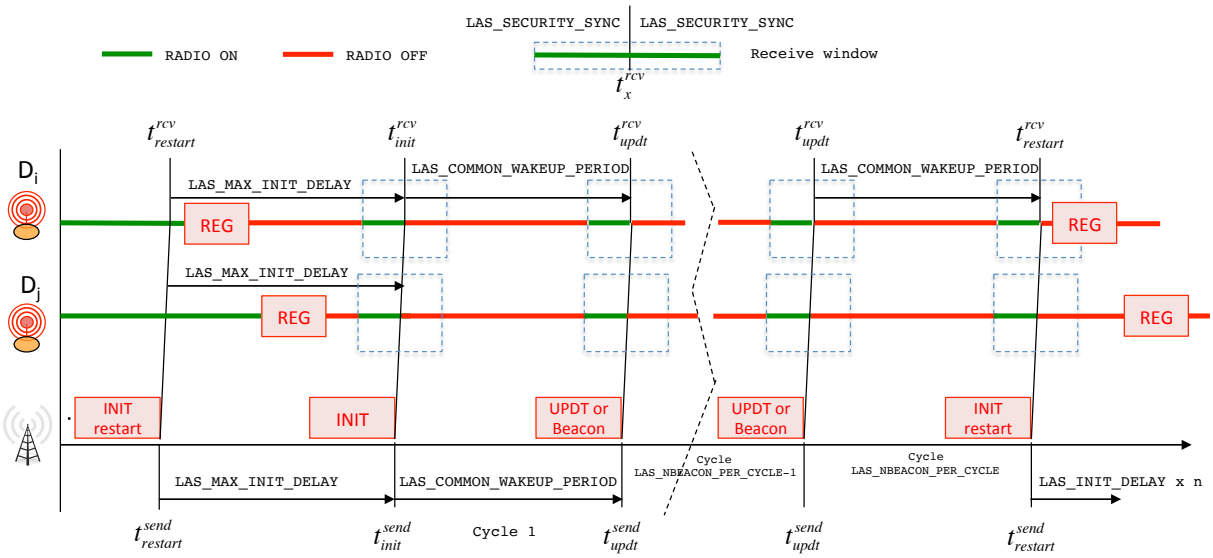

Figure 17 – INIT and UPDT message synchronization

When the devices received the INIT message, they start a cycle with all activity variables reset to their initial value. As mentioned previously, UPDT messages are also sent by the LR-BS in a synchronized manner: taking t_{init}^{send} as a reference time, the LR-BS sends an UPDT message every LAS_COMMON_WAKEUP_PERIOD. Similarly, an end-device can expect to receive such UPDT messages at $t_{init}^{rcv} + \text{LAS\_COMMON\_WAKEUP\_PERIOD}$. If there is noUPDT message to send, the LR-BS still send an UPDT message with $|AT^i| = 0$ and $D^i = 0$. Such a message with $|AT^i| = 0$ is interpreted as an UPDT Beacon message in order to periodically synchronize the end-devices: the next wake-up time for an end-device will be $t_{beacon}^{rcv} + \text{LAS\_COMMON\_WAKEUP\_PERIOD}$.

All devices are controlled by the LR-BS: an INIT message always initiates a new 1-hour cycle. It is preferable to choose LAS_COMMON_WAKEUP_PERIOD (expressed in ms) such as 3600000/LAS_COMMON_WAKEUP_PERIOD=LAS_NBEACON_PER_CYCLE is an integer number. When the LR-BS has sent LAS_NBEACON_PER_CYCLE−1, it will schedule at the next UPDT time an $\text{INIT}_{\text{restart}}$ message that will initiate a new cycle startup procedure with all active devices sending their REG messages and waiting for the INIT message, as previously described, to start a new cycle.

Figure 17 is not in scale and our implementation uses LAS_INIT_DELAY=2000ms and LAS_MAX_INIT_DELAY=LAS_INIT_DELAYxLAS_MAX_DEV. For 10 devices for instance, the LR-BS have 20s to receive all the REG messages before issuing the INIT message. We set LAS_COMMON_WAKEUP_PERIOD=300000ms (5min) and LAS_SECURITY_SYNC=2000ms. All values are expressed in milliseconds. In a 1-hour cycle, there are therefore LAS_NBEACON_PER_CYCLE−1=11 UPDT or Beacon messages, the last control message of the cycle being a new $\text{INIT}_{\text{restart}}$. With this simple synchronization method, end-devices can switch off their radio most of the time and remain synchronized for UPDT messages. We ran our test-platform with 3 image sensors for several hours without losing synchronization thanks to the Beacon messages that limit the clock drift to only a LAS_COMMON_WAKEUP_PERIOD.

Note that in Figure 17, an end-device must switch on its radio on startup in order to be able to receive the first $\texttt{INIT}_{\texttt{restart}}$ message. Actually, its radio must be on for at least an LAS_COMMON_WAKEUP_PERIOD duration. If an UPDT or Beacon message is received instead of the $\texttt{INIT}_{\texttt{restart}}$ message, the device will periodically wake-up to wait for the next $\texttt{INIT}_{\texttt{restart}}$ message. Meanwhile, it can only use its local activity time (i.e. 36s) before joining the LAS pool of devices or ask to be dynamically inserted into the running pool as explained in the next subsection.

2.4.4. Building and queueing UPDT messages

As mentioned previously, the current implementation uses the "All Device" to distribute remote activity usage on all the devices of the running pool and UPDT messages are sent by the LR-BS at specific moment to handle sleep periods at end-devices. With the "All Device" method, it is possible to reduce the LR-BS radio activity time for UPDT message transmission to a minimum by exploiting their cumulative behavior.

For instance, at the end of a transmission from device D_k, if a regular UPDT message $(|AT^k|, D_k)$ needs to be sent, the LR-BS actually only set the needUpdate flag in the device table for device D_k. The checkCycle() function will later call a checkUpdt() function when it is time to send UPDT messages. checkUpdt() will build and send UPDT messages for all devices in the device table with the needUpdate flag, if any (otherwise a Beacon will be sent instead). The value for $|AT^k|$ is simply $|AT^k| = |l^k_{RAT0} - lastl^k_{RAT0}|$ as performed by action 3b. As $lastl^k_{RAT0}$ is not updated until an UPDT message is built, we actually have the cumulative behavior described previously in subsection 3.4 and 3.5. Now, if an UPDT w/RATU message needs to be sent (i.e. $l^k_{RAT0} < 0$) and $lastl^k_{RAT0} > 0$, then the UPDT w/RATU message is actually built and queued for later transmission. By building the UPDT w/RATU message $lastl^k_{RAT0}$ is updated and the {\tt needUpdate} flag is cleared. If $lastl^k_{RAT0} < 0$, meaning that D_k already made usage of remote activity time, then the LR-BS actually set (or re-set) the needUpdate flag similarly to the regular UPDT case because in this case the UPDT w/RATU message has a cumulative behavior. In this way, the LR-BS can optimize its radio activity time by using cumulative behavior every time it is possible.

If we take the UPDT message of Figure 6, the current implementation delays the transmission of the UPDT by simple setting the needUpdate flag for device D_4. The second UPDT w/RATU message of Figure 7 is built and queued for transmission. By building the UPDT w/RATU message, $lastl^4_{RAT0}$ is updated and the needUpdate flag, that was previously set, is cleared. The third UPDT w/RATU message finds $lastl^4_{RAT0} < 0$ so the needUpdate flag is set again. Assuming that D_4 needs to send more packets, the next UPDT w/RATU can simply set again the needUpdate flag. When it is time to send the UPDT messages, all UPDT messages queued for transmission will be sent first, and applied sequentially by end-devices following action 4 or 5. Then, for all devices with the needUpdate flag set in the device table, the corresponding UPDT message is built and sent. This is illustrated in Figure 18 below.

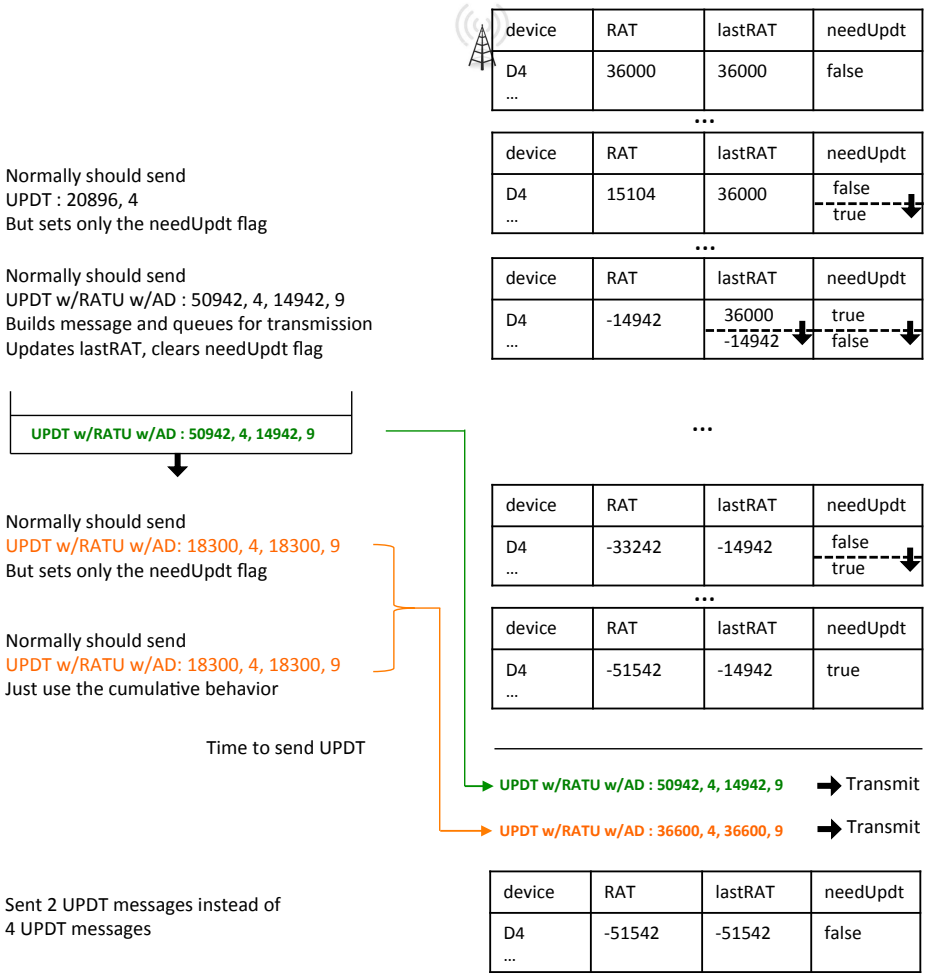

Figure 18 – Building and queuing UPDT messages

2.4.5. Dynamic insertion of new devices

Several devices D_k can be outside the pool of LAS devices for various reasons (reboot, maintenance, new deployment, late response, ...). They can ask to be inserted into the running LAS pool. In order to do so they can asynchronously send a $REG(l^k_{RAT0})$ message at any time, between 2 UPDT or Beacon messages, assuming, as indicated above, that each device can wake-up at appropriate moment to receive these messages. On reception of the REG messages, that have been sent without a prior $\texttt{INIT}_{\texttt{restart}}$ message, the LR-BS can allow the new devices to join the existing LAS cycle by responding with a dedicated UPDT message, noted $\texttt{UPDT}_{\texttt{adddev}}$, at the next UPDT message synchronization moment. The $\texttt{UPDT}_{\texttt{adddev}}$ message is sent after all UPDT messages scheduled for transmission and will start with $|AT^i| = 0$ and $D_i = 0$. If there is no UDPT message but only the Beacon, then this dedicated UPDT message will play the role of the Beacon as well since $|AT^i| = 0$ and $D_i = 0$. However, as opposed to the UPDT Beacon, the $\texttt{UPDT}_{\texttt{adddev}}$ message continues with l_{RAT0}, a number of new devices, n_d, and the list of new devices, $E\{D_k\}$. For instance, if only one new device D_k is added, then the $\texttt{UPDT}_{\texttt{adddev}}$ message has the following content: $\texttt{UPDT}_{\texttt{adddev}}$ $(0,0, l_{RAT0}, 1, E = \{D_k\}, G_{AT})$. The last field is the new G_{AT} value computed by the LR-BS. G_{AT} is actually the sum of all positive l^j_{RAT0} in the LR-BS's device table: $G_{AT} = \sum_{j=1}^{n}(l^j_{RAT0} : l^j_{RAT0} > 0)$. After sending the $\texttt{UPDT}_{\texttt{adddev}}$, LR-BS can assume that $\texttt{n=n+1}$ (or

n=n+ n_d in the general case) in order to increase the INIT_DELAY timer for the next $\text{INIT}_{\text{restart}}$ message. The LAS service can therefore be enhanced with action 6 as follows:

Action 6 – Device D_j receiving an $\text{UPDT}_{\text{adddev}}$ $(0, 0, l_{RAT0}, n_d, E\{D_k\}, G_{AT})$ from LR-BS

a) For each $D_j \notin E\{D_k\}$, takes the advertised l_{RAT0} and updates its local $G_{AT}^j = G_{AT}^j + n_d \times l_{RAT0}$ to take into account the additional activity time shared by new devices in $E\{D_k\}$.
b) For the new $D_j \in E\{D_k\}$, sets $G_{AT}^j = G_{AT} + n_d \times l_{RAT0}$.

Action 6a extends for all D_j NOT in the device list their local G_{AT}^j by $n_d \times l_{RAT0}$ (normally $n_d \times 36000ms$) while action 6b assigns for each new devices the current G_{AT} plus n_d local activity time, i.e. $n_d \times 36000ms$. Note that if one device D_k indicates in the REG message an $l_{RAT0} < 36000ms$ then a dedicated $UPDT_{adddev} = (0,0, l_{RAT0}^k, 1, E = \{D_k\}, G_{AT})$ needs to be sent specifically for D_k.

2.4.6. Summary of extended data packet format

We show in Figure 19 the additional packet format for the $\text{INIT}_{\text{restart}}$, the UPDT Beacon and the $\text{UPDT}_{\text{adddev}}$ messages sent by the LR-BS.

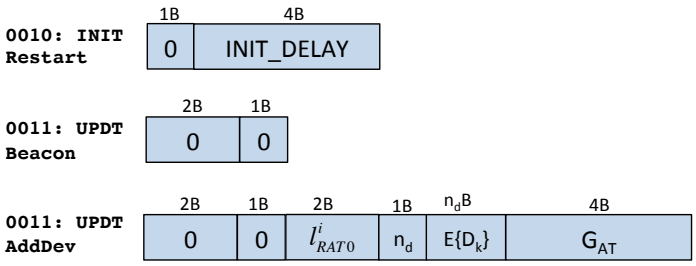

Figure 19 – Extended data packet format

2.5. Conclusions

In this chapter, we described a centralized activity time sharing mechanism for long-range unlicensed communications. The proposed mechanism targets scenarios where a pool of devices is deployed by a single organization and manages the individual activity time of all deployed devices in a shared manner. The objective is to allow a device to "exceptionally" transmit beyond the 1% duty-cycle limit in order to provide quality of service guarantees. Our proposition maintains the device-gateway system's consistency while keeping the number of additional control messages to a minimum. We described the proposed long-range activity sharing services with detailed examples highlighting how real-world deployment constraints are taken into account: (a) synchronization for network startup, (b) support for sleep period, (c) increase reliability with simple CSMA-like channel access mechanism, (d) cumulative behavior for updates and (e) dynamic insertion of new devices. The proposition is implemented as a C library for both the device and the gateway side. We demonstrate the integration of the long-range activity sharing services into existing projects and showed experiments with our long-range image sensor platform for visual surveillance.

3. OPTIMIZED CARRIER SENSE

Given the incredible worldwide uptake of long-range LoRa networks for a large variety of innovative IoT applications and the high flexibility in deploying private ad-hoc LoRa networks, it is important to consider dense environments and to improve the robustness of LoRa transmissions with more advanced channel access mechanisms. In this chapter, we investigate how a Carrier Sense mechanism can be adapted to decrease collisions in LoRa transmissions for both short and long messages. We explain the various steps towards the proposition of a CSMA protocol adapted to LoRa networks. We show experimental results using our low-cost IoT LoRa framework to implement innovative long-range image sensor nodes.

3.1. Channel access in wireless networks

We focus on competition-based channel access mechanisms at the MAC layer where nodes compete to get the channel. In this category, random access protocols such as the early ALOHA and various variants of CSMA, are widely used in wireless networks because of their relative simplicity and their distributed operation mode that does not require coordination nor overwhelming signaling overheads. There has been a notable amount of research done on the performance of ALOHA and CSMA in wireless networks. It is beyond the scope of this document to go through all these contributions but interested readers can start with [26,27,28].

3.1.1. IEEE 802.11

Among many CSMA variants, the one implemented in the IEEE 802.11 (WiFi) is quite representative of the approach taken by most of the random access protocols with so-called backoff procedure. Figure 20 illustrates the IEEE 802.11 CSMA mechanism used in the basic Distributed Coordinated Function (DCF) mode which is the common operation mode of WiFi networks with a base station. In this basic mode, the optional RTS/CTS mode is not used. The basic DCF IEEE 802.11 CSMA/CA (Collision Avoidance) works as follows :

- A node senses the channel to determine whether another node is transmitting before initiating a transmission
- A node senses the channel to determine whether another node is transmitting before initiating a transmission
- If the medium is free for a DCF inter-frame space (`DIFS`) the transmission will proceed (green `DIFS`)
- If the medium is busy (red `DIFS`), the node defers its transmission until the end of the current transmission and waits for an additional `DIFS` before generating a random number of backoff slot time in the range *[0, W-1]*.
- The backoff timer is decreased as long as the medium is sensed to be idle, and frozen when a transmission is detected on the medium and resumed when the channel is detected as idle again for more than `DIFS`
- When the backoff reaches 0, the node transmits its packet
- The initial W is set to 1. W is doubled for each retry (exponential backoff) until it reaches a maximum value

The random backoff timer is applied after a busy channel because it is exactly in that case that the probability of a collision is at its highest value. This is because several users could have been waiting for the medium to be available again.

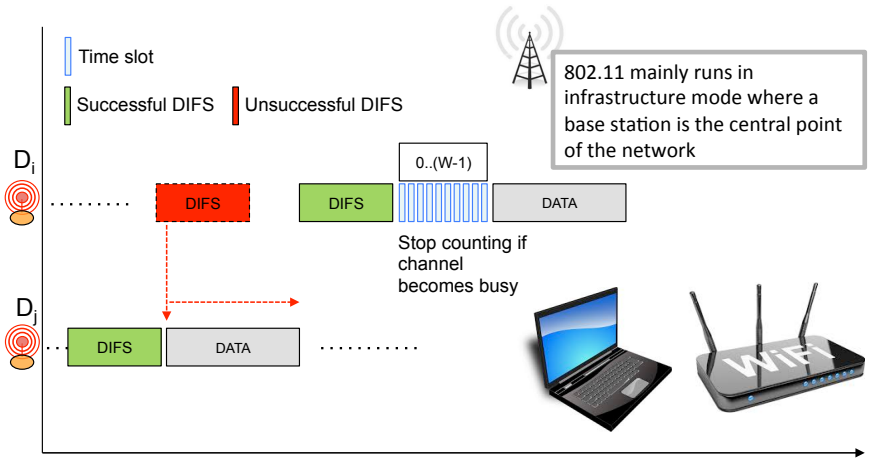

Figure 20 – IEEE 802.11 DCF CSMA/CA

3.1.2. IEEE 802.15.4

Closer to the domain of IoT, IEEE 802.15.4, that was very popular for wireless sensor networks (WSN), proposes both non-beacon-enabled mode with unslotted CSMA/CA channel access mechanism and beacon-enabled networks with slotted CSMA/CA. Here, we are describing the non-beacon-enabled mode as the beacon-enabled needs a coordinator and higher level of synchronization that is definitely not suited for LoRa IoT networks. %The IEEE 802.15.4 non-beacon-enabled with unslotted CSMA/CA mode works as follows:

Before a transmission, a node waits for a random number of backoff periods in the range $[0..2^{BE} - 1]$. BE is set to 3 initially

- If at the end of the waiting time the medium is sensed to be free the transmission will proceed
- If the medium is busy, the node defers its transmission, increases BE until it reaches a maximum value and waits for an additional $[0..2^{BE} - 1]$ backoff periods

Compared to IEEE 802.11, IEEE 802.15.4 always implements a backoff timer prior to any transmission and simply increases the backoff timer interval each time the channel is found busy for the same packet, without constantly checking the channel to know when it is going back to idle. There are several reasons for these differences. One reason is that simply increasing the backoff timer interval is less energy consuming than determining the end of the current transmission, especially if the transmission of a packet can take a long time (802.15.4 usually runs at 250kbps while 802.11 runs at 11Mbps and above). Another reason is that the node and traffic density of IEEE 802.15.4 networks is expected to be much smaller than those of WiFi networks. There is an additional reason 802.15.4's CSMA is different from 802.11's CSMA: 802.15.4 for WSN mainly runs under mesh topology (i.e. P2P and without central coordinator) with a shorter radio range (i.e. low transmit power), therefore the spatial reuse is higher, contributing again to decrease the traffic density at any given point in the network.

Again, there has been a huge amount of research in improving the basic 802.15.4 MAC protocol to better support multi-hop and duty-cycled low-power WSN. The reader can refer to [29] for a survey of MAC protocols for WSN.

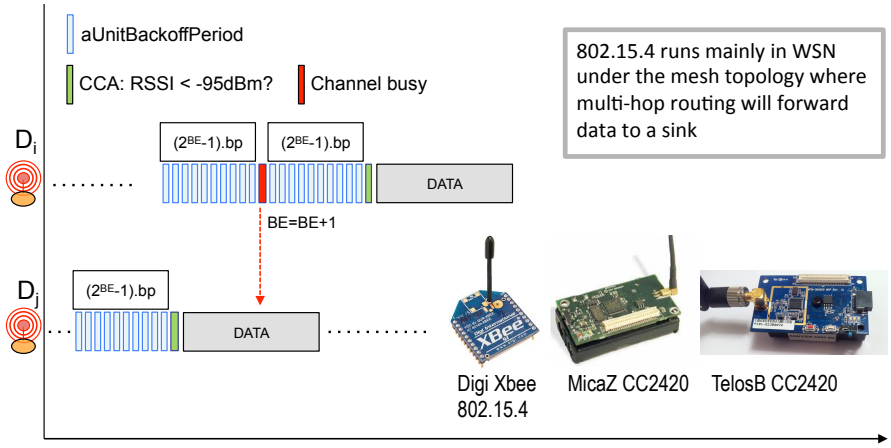

Figure 21 – IEEE 802.15.4 non-beacon unslotted CSMA

3.2. What can be done for LoRa

3.2.1. LoRa's channel activity detection (CAD)

Before investigating what CSMA approach can be adapted for LoRa, it is necessary to know how a LoRa channel can be defined busy or idle to implement a CS mechanism. As LoRa reception can be done below the noise floor the use of the RSSI is not reliable enough. For clear channel assessment, there is a special Channel Activity Detection (CAD) procedure that can be realized by a LoRa chip. We use the dedicated Arduino Due device to constantly perform CAD procedure and the interactive device to send periodic messages (see previous Figure 21). Figure 22 shows 2 cases: *(i)* 44 bytes message (40 bytes payload + 4 bytes header) every 15s with a CAD procedure every 100ms and *(ii)* 244 bytes message (240+4) every 15s with a CAD procedure every 1000ms. As can be seen in Figure 22 the LoRa CAD procedure can correctly detect all the LoRa transmission, and not only the preamble.

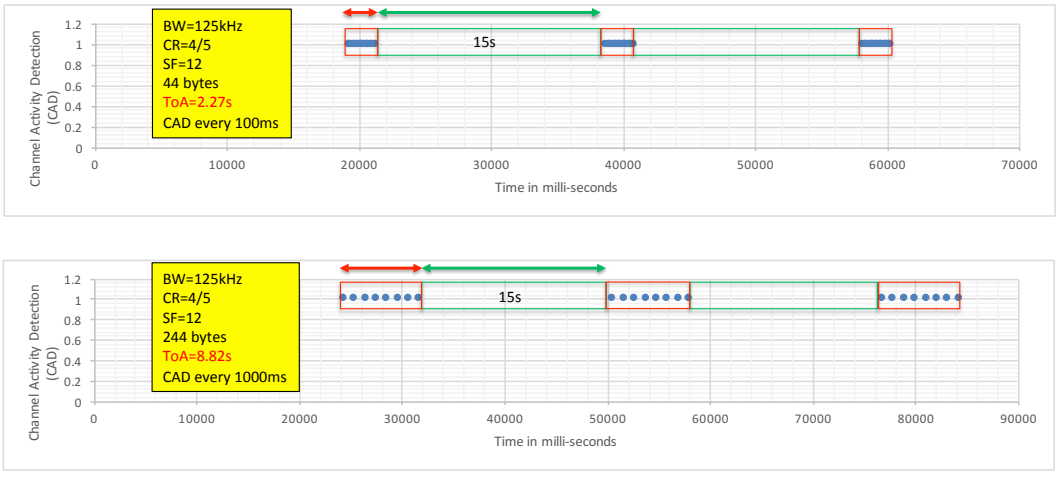

Figure 22 – Test of the LoRa CAD mechanism

3.2.2. Adaptation from 802.11

As a first attempt towards a CSMA protocol for LoRa, we start by adapting the previously shown 802.11 CSMA protocol and not the 802.15.4 one, although 802.15.4 is widely used in WSN and early IoT implementation, for 2 reasons. The first reason is that LoRa network architecture is mainly a single-hop star topology from devices to gateway, which is very similar to the WiFi topology with a base station. Therefore, the concept and the management of the 802.11's random backoff timer after a busy channel looks efficient for such environment. The second reason for not starting from 802.15.4 comes from its initial random waiting without channel sensing method that is more suitable for low-density networks than for high-density networks that will definitely be the case for LoRa networks.

To adapt the 802.11 CSMA protocol, we first need to define how the *DIFS* operation can be implemented. Usually, *IFS* should be related somehow to the symbol period T_{sym}. For LoRa, T_{sym} depends on BW and SF as follows: T_{sym}=2^{SF}/BW. For instance, LoRa mode 1 use BW=125kHz and SF=12 therefore T_{sym}^{mode-1}= 2^{12}/125000=0.032768. In [21], it is reported that the CAD duration is between 1.75T_{sym} and 2.25 T_{sym} depending on the spreading factor, see Figure 23. We performed some experimental tests to verify the real CAD duration against what is given in [21]: Figure 23 also shows the minimum and the maximum values measured with a 1ms-accuracy clock (the Arduino *millis()* function). We can see that the measured CAD durations are quite consistent.

LoRa mode	Tsym (ms)	CAD duration (Tsym)	CAD duration (ms)	Experimental measures	
				min value	max value
1	32.768	1.86	60.948	60	62
2	16.384	1.86	30.474	29	31
3	8.192	1.77	14.500	14	16
4	8.192	1.86	15.237	15	16
5	4.096	1.77	7.250	7	8
6	4.096	1.81	7.414	7	9
7	2.048	1.75	3.584	3	5
8	1.024	1.75	1.792	1	3
9	0.512	1.79	0.916	1	1
10	0.256	1.92	0.492	0	1

Figure 23 – Theoretical CAD duration and experimental measures

In our current implementation *DIFS* does not depend directly on T_{sym} but on the duration of the CAD mechanism, therefore, we assign an integer number of CAD to *DIFS*. Our communication library provides a low-level `doCAD(counter)` function that takes an integer number of CAD, i.e. `counter`, performs sequentially the requested number of CAD and returns 0 if all CAD have been successful (no channel activity). If one CAD detects activity the function exits with value greater then 0. The *DIFS* procedure shown in Figure 24 works that way and once a failed CAD has been observed the node exits the *DIFS* procedure and continuously checks for a free channel.

In Figure 24, *DIFS* is assigned 9 CAD which gives a duration of about 9 x 61ms=549ms for LoRa mode 1. At this point of the study, the duration of *DIFS* is not really important as we only need to be able to assert a free channel for a given duration. The value of 9 CAD provides enough time to detect channel activity and also provides the possibility to define a much shorter timer (using 3 CAD for instance), such as the 802.11's *SIFS* to implement priority schemes is needed, and still be able to detect channel activity. Then the random backoff timer is also defined as a number of CAD because the channel should be checked in order to freeze or continue the decrease of the backoff timer. The upper bound, *W* of the random backoff timer can be set in relation to the number of CAD defined for *DIFS*. For instance, if *DIFS=9* CAD then *W* can be defined as *n x DIFS*. For instance, if *n = 2* then *W=2x9=18* CAD.

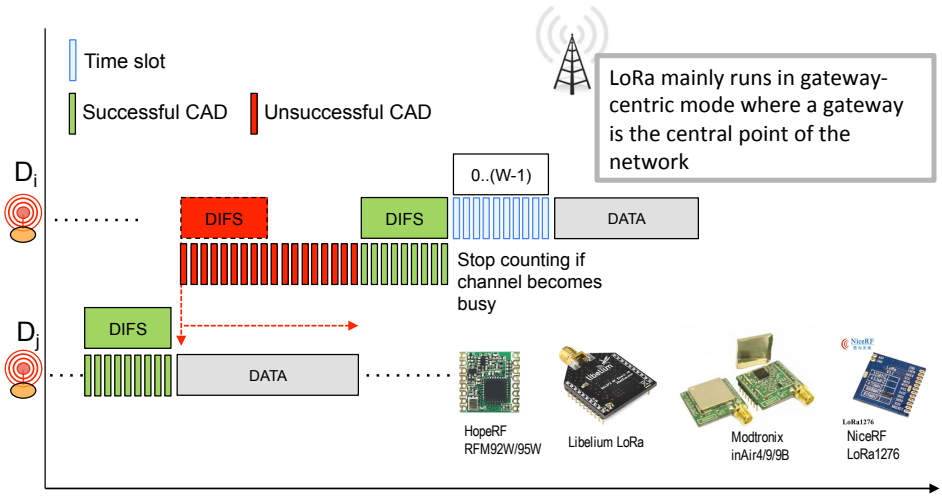

Figure 24 – CSMA mechanism adapted from IEEE 802.11

It is also possible to double *W* for each retry (exponential backoff) until it reaches a maximum value. However, while 802.11 initiates a retry when no ACK is received after a given time, the usage of acknowledgement is not common in LoRa as it is very costly for the gateway (the gateway is considered as a normal node and therefore its radio duty-cycle can be limited by regulations). Therefore, there is no such retry concept with unacknowledged transmissions. Nevertheless, when 802.11 doubles *W* for each retry the underlying assumption for the transmission errors is a denser channel. Here, we can follow the same guideline and double *W* each time the channel cannot be found free for an entire *DIFS*, starting from the second *DIFS* attempt. In the current implementation, we set *W=18* CAD initially and we can double it 3 times so the maximum value is *W*=144 CAD which will give a maximum wait timer of 8784ms for LoRa mode 1. If we add the value of the successful *DIFS* which is 9 CAD, i.e. 549ms, then the maximum total wait timer after a busy channel is about 9333ms which correspond roughly to the ToA of the maximum LoRa packet size. This property remains roughly true for all the defined LoRa modes and therefore can avoid waiting longer than necessary.

Figure 25 shows an experiment with the image sensor sending 4 image packets (about 240 bytes per packet) while the interactive device is sending medium-size messages of 40 bytes. The output is from the interactive device and it can be seen that the adapted CSMA protocol can nicely avoid the collision by deferring the transmission of the interactive message. In the

illustrated experiment, transmission is deferred only once before transmission succeeds as the time between 2 image packets is greater than a $DIFS$ plus the random backoff timer of 17 CAD.

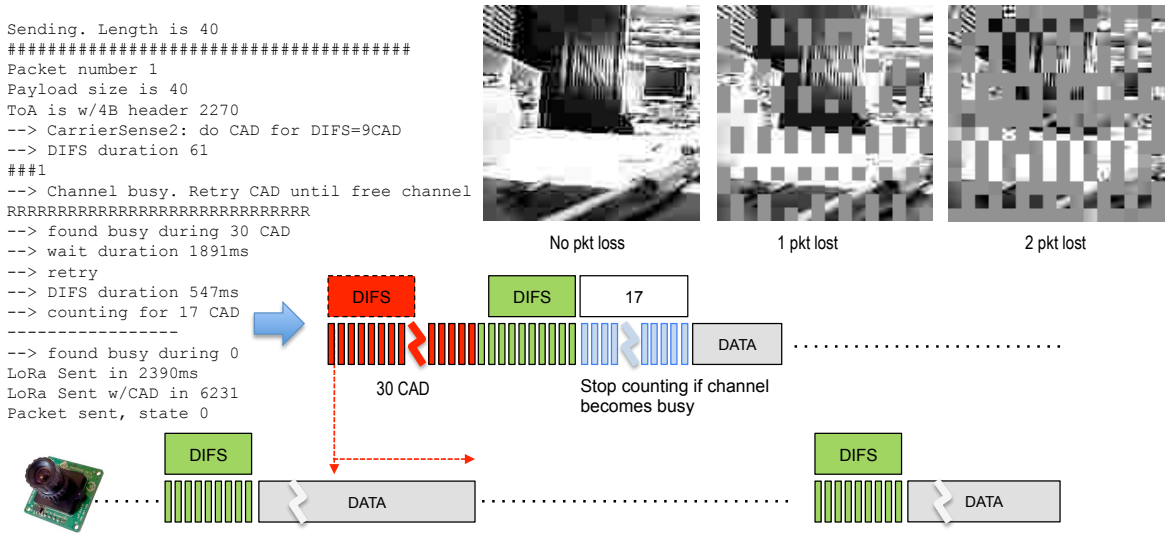

Figure 25 – Experimental test of the proposed CSMA adaptation

Figure 25 also shows the received image without any packet loss and 2 examples of received images when there is no channel access mechanism (pure ALOHA). It all our tests, the proposed CSMA protocol adapted from 802.11, and further referred to as $CSMA_{802.11}^{LoRa}$, totally avoids packet losses for both the image sensor and the interactive device.

3.2.3. CAD reliability issues

By testing further, the CSMA mechanism in various long-range deployment, we observed a fast decrease of the CAD's reliability when distance increases: although a transmission can be successful at several kilometers, CAD starts to not reliably detect the whole transmission when the distance to the sender is about 1km (with dense vegetation, CAD reliability can start to decrease even at 400m). Figure 26 shows CAD reliability with the same traffic pattern previously shown in Figure 22 but with the sender and the Arduino Due device performing CAD separated by 400m with some trees between them. As can be seen, the CAD procedure fails to detect channel activity many times during an on-going transmission.

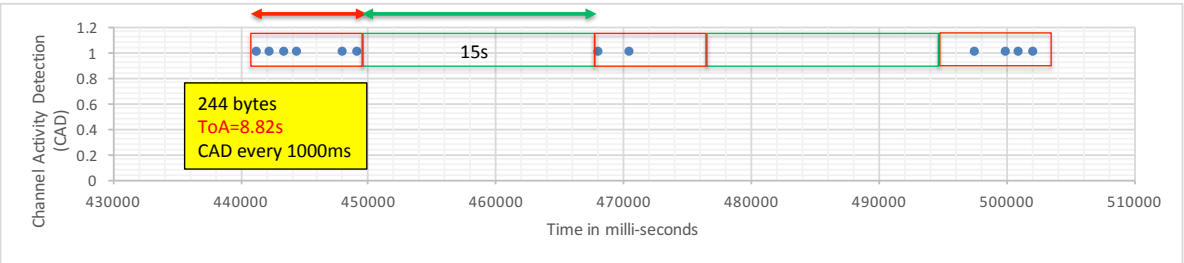

Figure 26 – CAD fails to detect activity of on-going transmissions

This CAD unreliability issue in real-world deployment scenario has a huge negative impact on the CS mechanism. For instance, in the previously proposed CSMA adaptation from 802.11, it is not possible anymore to rely on CAD to detect when the channel will become really free

after a busy state nor to rely on a successful $DIFS$ as a free channel indication to start transmission. However, what can be observed in Figure 22 and verified by the tests that we performed, is that during a long transmission the probability that all CAD attempts fail is quite low. In all our tests, and up to 1km in NLOS conditions, there has always been some successful CAD during any transmission.

3.2.4. Proposed CSMA mechanism

The CAD reliability issue raised previously calls for a different approach to prevent collisions. First, the previous $DIFS$ is extended to the ToA of the longest LoRa packet in a given LoRa mode, e.g. 9150ms for 255 bytes in LoRa mode 1. During this extended $DIFS(ToA_{max})$, CAD procedure is performed periodically (for instance every 1000ms as in Figure 22 (bottom). The purpose of $DIFS(ToA_{max})$ is to maximize the probability to detect an on-going transmission which can possibly be a long message with many unsuccessful CADs, thus appearing by mistake as a short message.

Then, when a CAD fails during a $DIFS(ToA_{max})$, instead of continuously waiting for a free channel followed by a $DIFS$+random backoff timer where CAD is checked constantly; here, there is a simple constant waiting period (pure delay) of ToA_{max}. Again, the purpose of the constant delay of ToA_{max} is to avoid performing CAD and transmission retries during the transmission of a possible long message, as a successful CAD does not guarantee a free channel. After the delay, the transmitter will try again to see a free channel for at least a $DIFS(ToA_{max})$ and the process continues until a maximum number of retries have been performed.

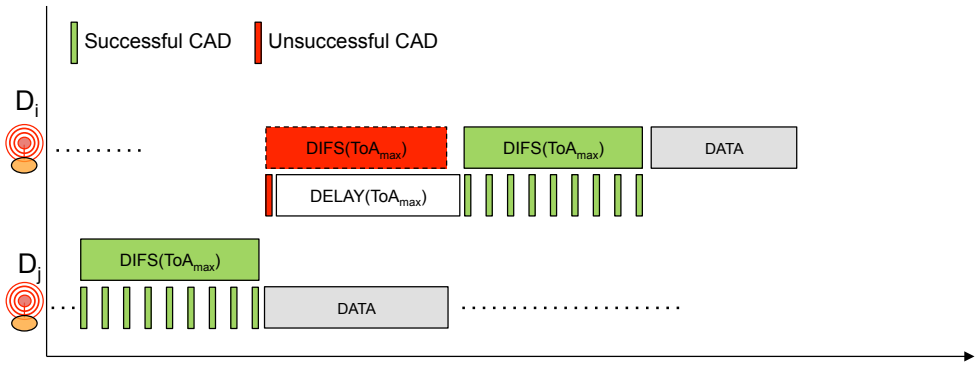

Figure 27 – New CSMA proposition

In all our tests with the new proposed CSMA protocol, noted $CSMA_{new}^{LoRa}$, we totally avoid packet losses for both the image sensor and the interactive device even when the nodes are hundredth of meters away from each other.

3.2.5. Discussions

3.2.5.1. CAD frequency during DIFS(ToA_{max})

A CAD procedure takes between 0.5ms and 61ms, from mode 10 down to mode 1, as shown in Figure 23 while the ToA of the longest LoRa packet, ToA_{max}, is respectively between 100ms and 9150ms as shown in Figure 20. Therefore, depending on the CAD failure probability (not detecting an on-going transmission) it is possible to increase or decrease the

number of CAD during a $DIFS(ToA_{max})$ to ensure at least 1 successful CAD to detect an on-going transmission. In our tests, we set the number of CAD to 9, similar to the number of CAD defined for a $DIFS$ in section B. Therefore the time between 2 CAD is $ToA_{max}/(9-1)$. For instance, in LoRa mode 1 where $ToA_{max}=9150ms$, there will be one CAD every 1143ms.

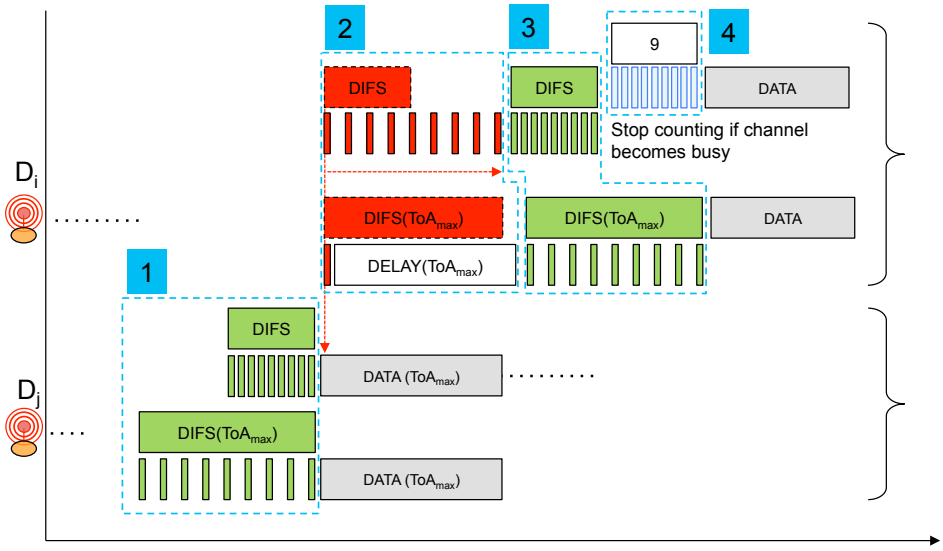

Figure 28 – Scenario for comparing $CSMA_{802.11}^{LoRa}$ and $CSMA_{new}^{LoRa}$

3.2.5.2. *Energy considerations*

We can compare the energy consumption between $CSMA_{802.11}^{LoRa}$ and $CSMA_{new}^{LoRa}$ with the scenario depicted in Figure 28: a long packet is transmitted by device j after a successful $DIFS$ and there is an attempt from device i right at the beginning of this transmission. In Figure 28 there are 2 lines for each device, the first line shows $CSMA_{802.11}^{LoRa}$ while the second line shows $CSMA_{new}^{LoRa}$. To perform the energy comparison, we measured for the Arduino Pro Mini and the Teensy32 the drawn current when performing CAD, when waiting using the $delay()$ function and when waiting using deep sleep (DS) mode: Arduino Pro Mini (CAD: 12mA; $delay( )$: 5.7mA; DS: 54uA); Teensy32 (CAD: 36mA; $delay( )$: 29.5mA; DS: 110uA). As expected, deep sleep mode provides a very low energy consumption compared to the $delay( )$ function and CAD operation. Therefore, it is possible to state that E_{DIFS}=$E_{DIFS(ToA_{max})}$= 9 x E_{CAD} . With this approximation, sensing for a free channel before transmission at device j - block 1 - has comparable energy consumption level in $CSMA_{802.11}^{LoRa}$ and $CSMA_{new}^{LoRa}$.

Then, for device i , with $CSMA_{802.11}^{LoRa}$, checking until the end of the transmission - block 2 - can be comparable a $DIFS(ToA_{max})$ with periodic CAD performed 9 times. Therefore the energy consumption can be approximated again to 9 x E_{CAD}. With $CSMA_{new}^{LoRa}$, $DIFS(ToA_{max})$ fails at the first CAD to continue with $DELAY(ToA_{max})$ which has negligible energy consumption using deep sleep mode for the waiting. Therefore, block 2 for $CSMA_{new}^{LoRa}$ has an energy consumption of 1 x E_{CAD}.

Block 3 for both CSMA protocols is comparable to block 1. Then, for device i with $CSMA_{802.11}^{LoRa}$ there is the random backoff timer - block 4. Assuming that the channel is always free for the pending transmission then the mean timer value is $W/2$. As W is initially set to

18 CAD then the random backoff timer has a mean duration of 9 CAD, thus an energy consumption of $9 \times E_{CAD}$.

Finally, for the scenario depicted in Figure 28, $CSMA_{802.11}^{LoRa}$ has a total energy consumption of $4 \times [9 \times E_{CAD}]$ while $CSMA_{new}^{LoRa}$ has an energy consumption of $2 \times [9 \times E_{CAD}] + 1 \times E_{CAD}$ which is about half the energy consumption of $CSMA_{802.11}^{LoRa}$ - exactly $36/19$ time less. If the channel is found busy in block 3, then block 2 is repeated N times with an energy consumption ratio of 1: 9 for $CSMA_{new}^{LoRa}$. Thus, in "heavy" traffic load, $CSMA_{new}^{LoRa}$ definitely shows a much lower energy consumption than $CSMA_{802.11}^{LoRa}$: $(3 + N) \times [9 \times E_{CAD}]$ for $CSMA_{802.11}^{LoRa}$ and $2 \times [9 \times E_{CAD}] + N \times E_{CAD}$ for $CSMA_{new}^{LoRa}$. With $N=2$ for instance, the ratio becomes $45/20$ which is now more than half.

If we take into account the CAD success probability (detecting an on-going transmission), noted $P_{CAD} =]0,1]$, then the total energy consumption for $CSMA_{new}^{LoRa}$ increases to $2 \times [9 \times E_{CAD}] + N \times \frac{1}{P_{CAD}} \times E_{CAD}$. Figure 29 shows the energy consumption when varying N and P_{CAD}. To get the real energy consumption, we have to multiply by the duration of a CAD in a given LoRa mode, see Fig. 23.

3.2.5.3. Latency

$CSMA_{new}^{LoRa}$ obviously increases the sending latency because $DIFS(ToA_{max})$ is much larger than $DIFS$ (9150ms compared to 549ms for LoRa mode 1 and 255 bytes messages). Also, instead of continuously checks for a free channel in block 2, the node attempting to transmit always waits for $DELAY(ToA_{max})$. However, as LoRa networks are mainly used for delay-tolerant IoT applications, we believe this latency issue has less importance than the capacity of limiting collisions in dense scenarios which was the primary design choice for $CSMA_{new}^{LoRa}$.

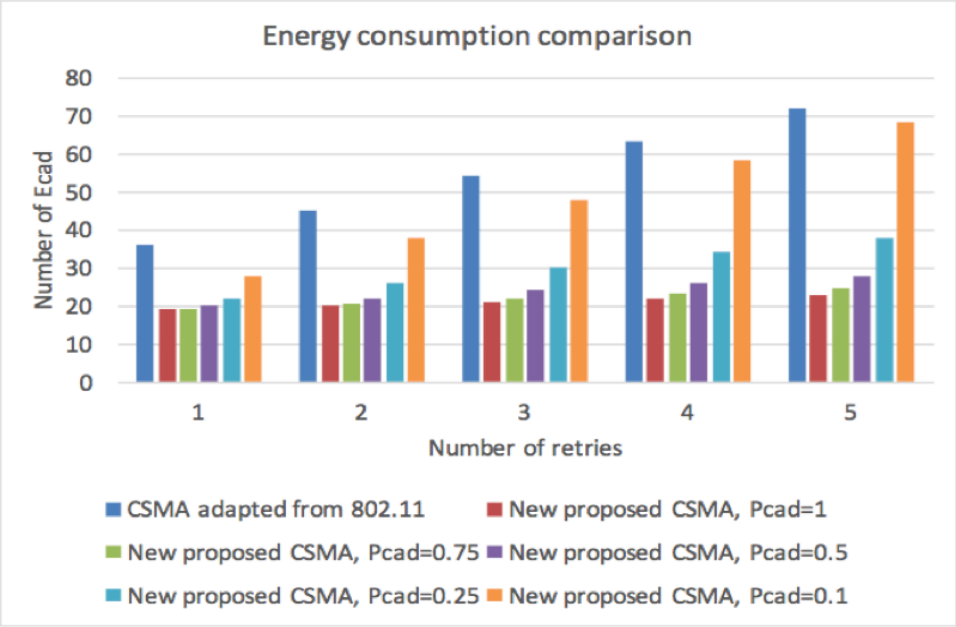

Figure 29 – Energy comparison of $CSMA_{802.11}^{LoRa}$ and $CSMA_{new}^{LoRa}$

3.2.6. Integration into the communication library

The communication library has been extended with these advanced CSMA mechanisms. There are currently 3 CSMA variants and they can be selected with the following function.

```
int8_t CarrierSense(uint8_t cs)
```

`CarrierSense(cs)` function performs a carrier sense procedure using the `cs` variant. `cs` can be 1, 2 or 3.

`cs` = 1 uses the CSMA procedure depicted in Figure 11. It was our first CSMA procedure proposed in the context of the LoRa Activity Sharing mechanism.

`cs` = 2 uses the CSMA procedure called $CSMA_{802.11}^{LoRa}$ and depicted in Figure 24.

`cs` = 3 uses the CSMA procedure called $CSMA_{new}^{LoRa}$ and depicted in Figure 27.

3.3. Conclusions

In this section, we investigated how a Carrier Sense mechanism can be adapted to decrease collisions in LoRa transmissions. We proposed a CSMA protocol adapted to LoRa networks, capable of handling both short and long messages. Experimental tests with image sensor nodes for innovative long-range image transmission showed very promising results where long on-going transmissions can be secured to avoid collisions even when the nodes are hundredth of meters away from each other. The proposed CSMA mechanisms have been integrated into the communication library and an end-device can run these mechanisms with dedicated functions to perform Carrier Sense prior to packet transmission.

4. 2-HOP LORA CONNECTIVITY

LoRa is designed for long-range communication where end-devices are directly connected to the gateway, which typically removes the need of constructing and maintaining a complex multi-hop network. Nevertheless, even with the advantage of penetration of walls, the range may not sometimes be sufficient. In this section, we investigate a 2-hop LoRa approach to extend the coverage area and solve these connectivity issues of real-world IoT deployment in rural environments. Very importantly, the main design objective is to propose a smart, transparent and battery-operated relaying mechanism that can be added after a deployment campaign to seamlessly provide an extra hop between the remote devices and the gateway. The remainder of the chapter is organized as follows. In Section 1 we outline related LPWAN technologies enabling multi-hop communication. Section 2 describes the proposed 2-hop LoRa approach based on low-power relay nodes. Performance evaluation and measurement results are discussed in Section 3.

4.1. Related work

In a multi-hop network, every node can communicate with the other nodes. They provide routing for each other so that two nodes physically far away from each other can communicate using nodes between them. Multi-hop alternatives for the uplink in LPWANs technologies have not been profoundly explored yet in networks operating at sub-1GHz. HARE protocol stack [30] is presented as a new LPWAN technology flexible enough to adopt multi-hop uplink communications when proving energetically more efficient. The gateway in HARE is responsible for controlling the stations (end-devices) by means of a beaconing schedule. This centralized approach allows stations to remain asleep the majority of their lifetime, waking up only to listen to beacons for receiving specific commands and/or configurations. HARE also includes a smart power management controller enabling stations to use the minimum power level required for reaching their next hop, which significantly reduces transmitting energy consumption.

In exploring the limits of LoRaWAN, the authors in [31] addressed the use of Time Division Access (TDMA) and multi-hop solutions in order to reduce both the number of collisions and the needed transmission power. From there, an extension of LoRaWAN protocol enabling relay-based communication to extend coverage area without the need of gateways and increase the performances of end-devices, is designed in [32]. This protocol allows end-devices to act as relay depending on the needs. Each Relay Eligible Node must periodically send a beacon to advertise itself to nearby end-devices. Every decision is taken by the network server, which sets up both the relays and the end-devices through MAC commands. In [33], the authors also analyzed the LoRa capacity and proposed LoRaBlink to support multi-hop communication where stations act as relays.

The purpose of this work is not to use the multi-hop concept to propose a new LPWAN protocol or an extension of LoRaWAN, to solve the aforementioned problems. In rural applications context for developing countries, gateways cannot act as relays as in [34] where more gateways are deployed to ensure multi-hop communication. This would lead to additional deployment cost since a gateway (a) is considered to be appropriately placed close to an unlimited power source, (b) requires an IP connection to operate and (c), is the

most expensive component, even in our low-cost context. End-devices also don't act as relays because they run very specific sensing template code and must be placed according to sensing needs.

4.2. Smart 2-hop relaying mode

4.2.1. Principle

Our 2-hop LoRa relay approach consists in a post deployment addition of an extra hop between some end-devices and the gateway. We propose to have ***relay-devices*** which are special low-power nodes different from the end-devices. However, similar to the end-devices, relay-devices are built from the generic hardware IoT platform but their unique feature is to extend the network coverage by performing data receive and forward operations. It does not take part in any data sensing, data processing nor aggregation tasks. One of the major considerations of a relay-device should be its appropriate location to cover areas where connectivity is either lost or unstable after the network deployment. We designed the relay-device with the following requirements:

- ***Low power***: relay-devices are battery-powered and therefore energy constrained. Their hardware should be very similar to those used by low-cost end-devices (i.e. Arduino Pro Mini). Being battery-operated they must not listen continuously, which basically would make them gateways and this is not what we wanted.

- ***Smart***: relay-devices must be designed to remain in low-power mode most of the time. Obviously, they have to wake-up at appropriate moments to catch uplink transmissions from specific devices in order to perform the relay operation. This is the major consideration of this work since missing uplink packets would make the network less reliable than it was. Therefore, a relay-device must be able to switch from sleep to active mode by smartly analyzing the uplink pattern from end-devices.

- ***Transparent***: relay-device nodes must be transparent to the rest of the network: (a) no change in hardware or software for end-devices or gateway to support the new 2-hop approach; (b) no additional signaling traffic between relay-devices and end-devices or gateway. Therefore, end-devices should not be aware of the 2-hop relay mode, nor to perform any discovery and binding process to a nearby relay-device. A relay-device also does not need to exchange parameters with the gateway for advertising its presence. And, on the gateway side, no scheduling mechanism for end-devices and relay-devices is required. The presence of a relay-device should not be detected although it is possible to indicate its presence with a specific flag in the packet header if it is desirable for the gateway (or network server) to have this information. Our approach is not centralized, neither at gateway nor network server as in related works. Furthermore, withdrawal or failure of a relay-device leaves the network as functional as before its integration in the network.

Figure 30 depicts our proposed architecture for providing a transparent 2-hop LoRa connectivity. The red link means no direct connectivity while the orange link means unstable connectivity. The green links means high quality, stable connectivity.

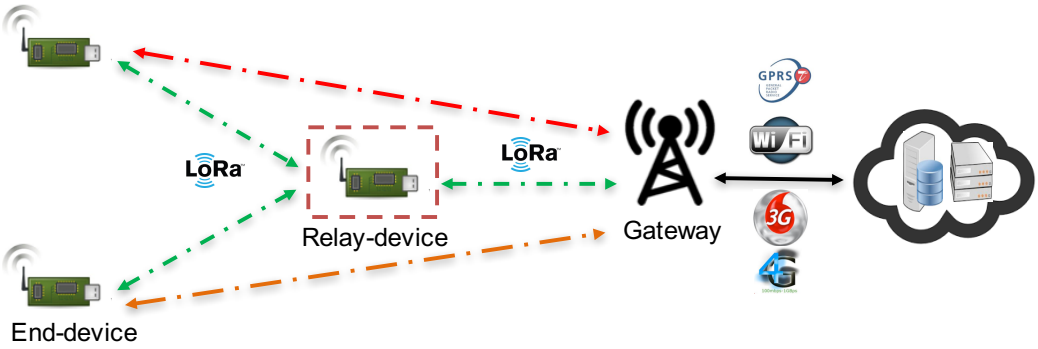

Figure 30–Long-range 2-hop connectivity architecture

The main advantage of our smart relaying mode is related to the relay-devices' ability to adapt in complete autonomy and transparency to their deployment environment. This is realized with an autonomous and asymmetric synchronization approach. It does not require any time synchronization between the nodes, e.g. end-devices behavior remains unchanged as indicated previously. It is asymmetric in the sense that the synchronization work is done by the relay-device: only the relay-device has to learn wakeup periods of the end-devices.

4.2.2. Implementation

In a typical telemetry LoRa network, end-devices periodically measure environmental parameters and transmit data packets mostly at regular intervals, being most of the time in deep sleep mode where they are not able to send nor receive packets. We assume here that end-devices wake-up at least once every 60 minutes -- from their local time as there is no synchronicity between end-devices. When inserted in an existing LoRa network, relay-devices are responsible for forwarding data packets from end-devices with no prior knowledge of how end-devices will wake up. Once deployed, a relay-device discovers end-devices in its vicinity and will build a wake-up table. When powered-on a relay-device first runs an observation phase and then a data forwarding phase.

4.2.2.1. Observation stage

The observation phase consists in observing network traffic for a specified duration to learn the traffic pattern. At start-up a relay-device usually does not know when it will receive an uplink packet, so it needs to be in receive mode during all the observation duration. This observation duration must be long enough to catch the various uplink packets from end-devices. Assuming that end-devices wake-up at least once every 60 minutes, an observation duration longer than 60 minutes is sufficient. The Arduino Pro Mini running at 3.3v consumes about 15mA in continuous receive mode so 60 minutes of observation has little impact on the battery lifetime as this process is only performed on startup. In the observation phase a relay-device receiving an uplink packet from an end-device (a) sends to the device any cached downlink packet; (b) records relevant information of the uplink packet such as the source address, the timestamp, etc.; (c) forwards the packet to the gateway by keeping the original packet header. Note that packet forwarding from a relay-device to the gateway during this phase can also allow for transmission quality comparison if the original packet also reaches the gateway.

Note that the relay-device can also receive downlink packets from the gateway to specific devices. In this case, the relay-device stores this downlink packet and will forward it at the next uplink transmission from the corresponding end-device. The reason to do so, instead of directly forwarding the downlink packet to the device is because the device receive window probably does not take into account the additional delay introduced by the relay-device.

This process, detailed in Algorithm 1 is repeated throughout the observation duration. When the observation phase is over the relay-device switches to the data forwarding phase.

Algorithm 1 Observation stage

Input:

obs_duration: *appropriate time interval for observing the network traffic in order to synchronize with it.*
Bound_Devices: *an array of bounded devices to the relay device, sorted in receiving packet order.*

```
1:  while obs_duration ≥ current_time do
2:    pkt ← Receive_Pkt()
3:    if pkt ≠ Null then
4:      if pkt.Src ≠ Gateway_id then
5:        //it's an uplink data
6:        if not is_bound(pkt.Src) then
7:          //it's the first reception for this device
8:          Record_Device(Bound_Devices, pkt.Src, times-
            tamp)
9:        else
10:         /*send back downlink message to pkt.Src device,
            if present. Then delete it */
11:         Send_Back_Downlink_Msg(Bound_Devices,
            pkt.Src)
12:
13:         /*update pkt.Src device in Bound_Devices */
14:         Update_Device(Bound_Devices, pkt.Src, times-
            tamp)
15:       end if
16:       /*forward the received packet to the gateway by
          keeping the original packet header */
17:       Forward_Data(pkt)
18:
19:     else
20:       //it's a downlink message
21:       if is_bound(pkt.Src) then
22:         Store_Msg(Bound_Devices, pkt.Src, pkt.Msg)
23:       else
24:         Record_Device_Msg(Bound_Devices, pkt.Src,
            timestamp, pkt.Msg)
25:       end if
26:     end if
27:   end if
28: end while
```

4.2.2.2. Data forwarding stage

With the collected information during the observation phase, the relay-device is now able to determine wakeup time of the end-devices in its vicinity. It can determine its own activity schedule in each round to wakeup at appropriate moment to forward uplink packets and remain in low-power mode the rest of the time. Algorithm 2 shows how the sleep period is computed.

Algorithm 2 Compute next reception period

Input:

Bound_Devices: *an array of bound devices to relay device, sorted in receiving packet order.*

Output:

reception_time: reception period of the next packet

```
1: min_time ← Bound_Devices[0].timestamp + Bound_Devices[0].reception_interval
2: for i = 1 to Bound_Devices.size() do
3:    tmp_time ← Bound_Devices[i].timestamp + Bound_Devices[i].reception_interval
4:    min_time ← min(min_time, tmp_time)
5: end for
6: reception_time ← min_time − current_time
7: return reception_time
```

In the data forwarding phase the relay-device determines the wakeup time T (using *sleep_period*) to wake up to catch the next uplink packet from device i. The relay-device will actually wake up at $T - T_{TOLERANCE}$ in order to compensate for clock drift and $T_{TOLERANCE}$ must be kept small to reduce energy consumption. Once awake, the relay-device enters receive mode waiting for the next uplink packet until time $T + T_{TOLERANCE}$. When receiving the uplink packet, it simply forwards the packet to the gateway. If it receives a downlink message in the receive window, it stores the message until the next upstream transmission from the corresponding end-device as explained previously for the observation phase. Note that upon reception of the uplink packet from device i the relay-device updates the wakeup time of device i accordingly to take into account any clock drift. This process is repeated as illustrated in Algorithm 3.

Algorithm 3 Data forwarding stage

Input:

T_TOLERANCE: *appropriate time to wake up relay-device before the reception period of the next packet.*
Bound_Devices: *an array of bound devices to relay device, sorted in receiving packet order.*

```
1: while (1) do
2:   //reception period of the next packet
3:   reception_time ← Next_Recept_Period(Bound_Devices)
4:   idle_time ← reception_time − T_TOLERANCE
5:   waiting_time ← reception_time + T_TOLERANCE
6:
7:   /*switch to sleep mode, then wakeup at T_TOLERANCE time before next reception period */
8:   Go_To_Sleep(idle_time)
9:
10:  is_pkt_received ← false
11:  /*once awake, it waits for packet at most waiting_time period */
12:  while waiting_time ≥ time and not is_pkt_received do
13:    pkt ← Receive_Pkt()
14:    if pkt ≠ Null then
15:      /*send back downlink message to pkt.Src device, if present. Then delete it */
16:      Send_Back_Downlink_Msg(Bound_Devices, pkt.Src)
17:
18:      /*forward the received packet to the gateway by keeping the original packet header */
19:      Forward_Data(pkt)
20:
21:      msg ← Receive_Downlink_Msg()
22:      if msg ≠ Null then
23:        Store_Msg(Bound_Devices, pkt.Src, msg)
24:      end if
25:      /*update pkt.Src device in Bound_Devices */
26:      Update_Device(Bound_Devices, pkt.Src, timestamp)
27:
28:      is_pkt_received ← true
29:    end if
30:  end while
31:
32:  //timeout expired
33:  if not is_pkt_received then
34:    /*update timestamp of expected device*/
35:    Update_Timestamp(Bound_Devices, Expected_Dev, reception_time)
36:  end if
37: end while
```

4.3. Performance evaluation

We performed field tests to assess the performance of the proposed 2-hop approach for increasing network reliability. The university campus with many vegetation and sparse buildings has been our rural environment of deployment. We deployed 5 nodes: 3 end-devices ED_1, ED_2, ED_3), 1 relay-device (RD) and 1 gateway (GW). Both gateway and relay-device were placed in the second floor of two buildings. End-devices are soil humidity sensing devices located in different places at 20cm above the ground. End-devices have different transmission intervals: ED_1 sends a data packet every 13 minutes, ED_2 every 23 minutes and ED_3 every 29 minutes. LoRa parameters of the experiments were chosen as follows: spreading factor of 12, bandwidth of 125kHz and coding rate of 4/5, which is the usual setting that provides the longest range. The transmission power for all tests has been set to 14dBm and all measurements were done in non-LOS conditions.

Our first version of the relay-device using an Arduino Pro Mini suffers major clock drifts as the internal timer interrupt to measure time is not called when the relay-device is put in deep sleep mode. Once awake, the current time shows a clock drift roughly equal to the sleep duration, for instance 15 minutes. It is possible to compensate for this clock drift but the mechanism is approximative as the real sleep duration is not exactly the expected sleep duration. Since Arduino boards have no embedded Real-Time-Clock (RTC), we expand our relay-device with a small DS1307 RTC board to obtain the second version of the relay-device.

Our first tests confirmed what has already been validated in the related work: adding an extra hop between end-devices and gateway can (a) extend coverage area without the need of gateways, (b) significantly increase the link reliability in high packet error rate conditions. Therefore, our tests specifically validate the smart approach of the relay-device which automatically synchronizes the relay-device with the rest of the network. Two relevant metrics have been identified: the packet error rate and the energy consumption.

4.3.1. Percentage of correctly received packets

As the relay-device should wakeup in advance (by $T_{TOLERANCE}$) to safely catch the next uplink packet the value for $T_{TOLERANCE}$ is critical: a too small value may make the relay-device miss the uplink packet while a too large value would consume more energy. We varied $T_{TOLERANCE}$ from 0s to 5s and measured the ratio of correctly received packets at the relay-device. The observation phase is set to 60 minutes and at least two packets per end-devices are expected to be catched. We ran each test during 3 hours: the first hour for the observation phase and the last 2 for the data forwarding phase. Results are shown in Figure 31.

Of course, during the observation phase all packets sent by the end-devices are correctly received and forwarded to the gateway. We can see in Figure 31 that when $\mathrm{T_{TOLERANCE}} \leq 2s$ we have a total desynchronization of the relay-device with the rest of the network. This is mainly due to the fact that the wakeup of the relay-device takes some time. When $\mathrm{T_{TOLERANCE}} > 2s$, we have partial synchronization: at least 50% of packets can be correctly received but not more than 70%. This is due to a small clock drift. When $\mathrm{T_{TOLERANCE}} = 5s$, it is possible to obtain 100% of correctly received packets.

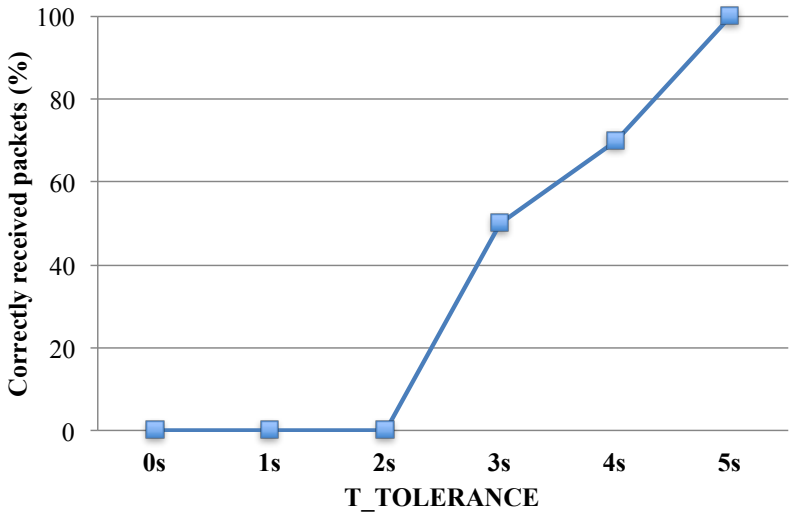

Figure 31– Correctly received packets at relay-device

4.3.2. Discussions on energy consumption

The Arduino Pro Mini with the LoRa module draws about 40mA when active (taking a measure) and transmitting. The whole process takes about 2s. In deep sleep mode, the board draws 5uA. Therefore, an end-device that sends 1 measure every hour consumes in the average $(2 * 40mA + 3598 * 0.005mA)/3600 = 0.0272mA$. We have real devices running on AA batteries that have been functioning for more than 2 years at time of writing.

Regarding the relay-device, it has to wakeup and forward uplink packets. For each wakeup, there will be a continuous receive during $2 * T_{TOLERANCE} = 10s$ at maximum. In receive mode, the board draws 15mA. Then, it has to forward the packet which has an energy consumption similar to the transmission from an end-device, i.e. 40mA during 2s. Therefore, for each uplink packet, the relay-device consumes in the average $(10s * 15mA + 2s * 40mA)/12\,s = 19.16mA$. If we assume that a relay-device is used to relay a very small number of isolated end-devices, e.g. 3 end-devices, then the number of wakeup can be limited. For instance, with 3 end-device, the relay-device has to wakeup 3 times per hour resulting in an average consumption of $(10s * 15mA + 2s * 40mA)/12\,s = 19.16mA$. If we assume that a relay-device is used to relay a $(3 * 12s * 19.16mA + (3600s - 3 * 12s) * 0.005)/3600\,s = 0.196mA$ which still allows for more than a year of operation. The energy consumption introduced by the RTC is very small as during sleep periods the battery backup mode is used which only draws 500nA according to the DS1307's specification sheet. In order to read the RTC for resynchronization purpose the DS1307 must be awaken only for a small amount of time.

4.3.3. Discussions on radio duty-cycle

The flexibility of long-range transmission in the unlicensed bands can come at the cost of stricter legal regulations such as limited duty-cycling, e.g. maximum transmission time per hour. For instance, in Europe, electromagnetic transmissions in the unlicensed EU 863-870MHz Industrial-Scientific-Medical (ISM) band used by Semtech's LoRa technology falls into the Short Range Devices (SRD) category. The ETSI EN300-220-1 document [3] specifies for Europe various requirements for SRD devices, especially those on radio activity. Basically,

a transmitter is constrained to 1% duty-cycle (i.e. 36s/hour) in the general case. This duty cycle limit applies to the total transmission time, even if the transmitter can change to another channel.

Obviously, a relay-device that has to forward uplink packets from n end-devices will have to transmit at least n packets/hour. Assuming that each transmission takes about 1.5s (approximatively the time-on-air of a 20-byte payload packet -- header included) then a relay-device can relay 24 packets/hour which is quite sufficient in most of the cases.

4.4. Conclusions

We described in this section a 2-hop LoRa approach to increase coverage in real-world deployment scenarios. We proposed a smart, transparent and low-power relay-device that can be added seamlessly into an existing LoRa network, between some end-device and the gateway. Both end-devices and gateway are unchanged and can work with or without the relay-device. The experimental tests demonstrate the effectiveness of our approach, especially validating the relay-device's ability to synchronize in an automatic and asymmetric way with the rest of the network. Using low-cost hardware for the relay-device, the experimental tests also show that a safety wakeup of 5s prior to the expected time of receiving an uplink packet is sufficient to significantly increase the network reliability.

5. CONCLUSIONS

This deliverable 2.2 entitled « Low-latency and low-energy MAC for IoT connectivity » advanced research activities contributing to achieving lower latencies in duty-cycle environments and lower power consumption by avoiding costly packet retransmissions and/or losses. All the outcomes have been integrated into the WAZIUP IoT communication library developped in WP2 and previously presented in D2.1.

These contributions have been published in:

C. Pham, "Investigating and Experimenting CSMA Channel Access Mechanisms for LoRa IoT Networks", Proceedings of the IEEE WCNC conference, Barcelona, Spain, April 15-18, 2018.

C. Pham, "Robust CSMA for Long-Range LoRa Transmissions with Image Sensing Devices", Proceedings of the 10th Wireless Days conference, Dubai, UAE, April 3-5, 2018

C. Pham and M. Diop, "Demo: Deploying Low-cost, Long-range Innovative IoT with the WAZIUP IoT/Gateway Framework", Demo at the EWSN Conference, Proceedings of the EWSN conference, Madrid, Spain, February 14-16, 2018.

C. Pham, "QoS for Long-Range Wireless Sensors under Duty-Cycle Regulations with Shared Activity Time Usage". ACM Transactions on Sensor Networks, Vol. 12(4), 2016.

C. Pham, "Towards Quality of Service for Long-range IoT in Unlicensed Radio Spectrum". Proceedings of Wireless Days (WD'2016), Toulouse, France, March 2016.

REFERENCES

[1] LoRa Alliance, 2015. LoRaWAN specification v1.0.(2015).

[1] P. Chen, P. Ahammad, C. Boyer, Shih-I Huang, Leon Lin, E. Lobaton, M. Meingast, Songhwai Oh, S. Wang, Posu Yan, A.Y. Yang, Chuohao Yeo, Lung-Chung Chang, J.D. Tygar, and S.S. Sastry. "CITRIC: A low-bandwidth wireless camera network platform". In *Distributed Smart Cameras, 2008. ICDSC 2008*. Second ACM/IEEE International Conference on. 1--10.

[3] ETSI. 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; Part 1: Technical characteristics and test methods. (2012).

[4] C. Goursaud and J.M. Gorce. "Dedicated networks for IoT : PHY / MAC state of the art and challenges". *EAI Endorsed Transactions on Internet of Things* 1 , 1 (2015).

[5] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. 2000. "Energy-efficient Routing Protocols for Wireless Microsensor Networks". In *Proc. 33rd Hawaii Int. Conf. System Sciences (HICSS)*.

[6] S. Hengstler, D. Prashanth, Sufen Fong, and H. Aghajan. "MeshEye: A Hybrid Resolution Smart Camera Mote for Applications in Distributed Intelligent Surveillance". In *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on.* 360--369.

[7] IBM. LoRa WAN in C. http://www.research.ibm.com/labs/zurich/ics/lrsc/lmic.html . (accessed 14/07/2015).

[8] Semtech Jeff McKeown}. "LoRa™ - A Communications Solution for emerging LPWAN, LPHAN and Industrial Sensing & IoT applications". http://cwbackoffice.co.uk/docs/Jeff20McKeown.pdf . (accessed 13/01/2016).

[9] H. Khanmirza, O. Landsiedel, M. Papatriantafilou, and N. Yazdani. "Evaluating passive neighborhood discovery for Low Power Listening MAC protocols". In *IEEE WiMob*, 2014.

[10] Vincent Lecuire, Leila Makkaoui, and Jean-Marie Moureaux. "Fast zonal DCT for energy conservation in wireless image sensor networks". *Electronics Letters* 48, 2 (2012).

[11] Libelium. 2014. Waspmote LoRa 868MHz 915MHz SX1272 Networking Guide. v4.0-11/2014. (2014).

[12] Libelium. "Extreme Range Links: LoRa 868/915MHz SX1272 LoRa module for Arduino, Raspberry Pi and Intel Galileo". http://www.cooking-hacks.com/documentation/tutorials/extreme-range-lora-sx1272-module-shield-arduino-raspberry-pi-intel-galileo . (accessed 13/01/2016).

[13] S. Paniga, L. Borsani, A. Redondi, M. Tagliasacchi, and M. Cesana. "Experimental Evaluation of a Video Streaming System for Wireless Multimedia Sensor Networks". In *Proceedings of the 10th IEEE/IFIP Med-Hoc-Net*, 2011.

[14] Congduc Pham. 2014. "Communication performances of IEEE 802.15.4 wireless sensor motes for data-intensive applications: A comparison of WaspMote, Arduino MEGA, TelosB, MicaZ and iMote2 for image surveillance". *Journal of Network and Computer Applications* 46, 0 (2014), 48 -- 59.

[15] C. Pham. 2015a. "Deploying a Pool of Long-Range Wireless Image Sensor with Shared Activity Time". In *11th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'2015).*

[16] C. Pham. 2015b. "Low cost wireless image sensor networks for visual surveillance and intrusion detection applications". In *12th IEEE International Conference on Networking, Sensing and Control (ICNSC).*

[17] C. Pham and V. Lecuire. 2015. “Building low-cost wireless image sensor networks: from single camera to multi-camera system”. In *9th ACM International Conference on Distributed Smart Cameras (ICDSC).*
[18] Congduc Pham, Abdallah Makhoul, and Rachid Saadi. “Risk-based adaptive scheduling in randomly deployed video sensor networks for critical surveillance applications”. *Journal of Network and Computer Applications* 34, 2 (2011), 783 -- 795.
[19] Joseph Polastre, Jason Hill, and David Culler. 2004.” Versatile low power media access for wireless sensor networks”. In *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys'04).* 95--107.
[20] Mohammad Rahimi, Rick Baer, Obimdinachi I. Iroezi, Juan C. Garcia, Jay Warrior, Deborah Estrin, and Mani Srivastava. 2005. “Cyclops: In Situ Image Sensing and Interpretation in Wireless Sensor Networks”. In *ACM SenSys*.
[21] Semtech. 2014. “SX1272/73-860 MHz to 1020 MHz Low Power Long Range Transceiver”. Rev.2-07/2014. (2014).
[22] Semtech. 2015b. “LoRa modulation basics”. Rev.2-05/2015. (2015).
[23] Semtech. accessed 14/07/2015a. LoRa MAC. https://github.com/Lora-net/LoRaMac-node. (accessed 14/07/2015).
[24] Mo Sha, Gregory Hackmann, and Chenyang Lu. 2013. “Energy-efficient Low Power Listening for Wireless Sensor Networks in Noisy Environments”. In *Proceedings of the 12th International Conference on Information Processing in Sensor Networks (IPSN '13).* ACM, New York, NY, USA, 277--288. DOI: http://dx.doi.org/10.1145/2461381.2461415 .
[25] Wei Ye, John Heidemann, and Deborah Estrin. “Medium access control with coordinated adaptive sleeping for wireless sensor networks”. *IEEE/ACM Trans. Netw*. 12, 3 (June 2004), 493--506.
[26] M. Kaynia and N. Jindal, “Performance of ALOHA and CSMA in spatially distributed wireless networks,” in *IEEE ICC’08*.
[27] Y. Yang and T.-S. P. Yum, “Delay distributions of slotted ALOHA and CSMA”, *IEEE Trans. Comm.*, vol. 51, 2003.
[28] F. A. Tobagi, “Distribution of packet delay and interdeparture time in slotted ALOHA and CSMA”, *J. Assoc. Comput. Mach.*, vol. 29, 1982.
[29] A. Bachir, M. Dohler, T. Watteyne, and K. K. Leung, “MAC essentials for WSN”, *IEEE Comm. Surveys and Tutorials*, 12(2), 2010.
[30] T. Adame, S. Barrachina, B. Bellalta, and A. Bel, “HARE: supporting efficient uplink multi-hop communications in self-organizing lpwans,” CoRR, 2017.
[31] F. Adelantado, X. Vilajosana, P. Tuset, B. Martinez, J. MELIA-SEGUI, and T. Watteyne, “Understanding the Limits of LoRaWAN,” IEEE Communications Magazine, 2017.
[32] A. Sanfratello, “Enabling relay-based communication in lora networks for the internet of things: design, implementation and experimental evaluation,” Master’s thesis, University of Pisa, Italy, 2016.
[33] B. Martin, V. John, and R. Utz, “Lora for the internet of things,” In Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN), pp. 361–366, 2016.
[34] D. Lundell, “Ad-hoc network possibilities inside lorawan,” Master’s thesis, Lund University, Sweden, 2017. [12] LoRa-Alliance, “Lorawan specification, v1.0.2,” 2016

ACRONYMS LIST

Acronym	Explanation
ABP	Activation by Personalization
AES	Advanced Encryption Standard
AFA	Adaptive Frequency Agility
API	Application Programming Interface
BW	Bandwidth
CSS	Chirp Spread Spectrum
DIY	Do-It-Yourself
DSSS	Direct Sequence Spread Spectrum
ETSI	European Telecommunications Standards Institute
FAQ	Frequently Asked Questions
FEC	Forward Error Correction
FHSS	Frequency Hopping Spread Spectrum
FSK	Frequency Shift Keying
GPRS	General Radio Packet Service
GPS	Global Positioning System
GSM	Global System for Mobile communications
IDE	Integrated Development Environment
IoT	Internet-of-Thing
IP	Internet Protocol
ISM band	Industrial Scientific Medical band
JSON	JavaScript Object Notation
LBT	Listen Before Talk
LOS	Line of Sight
LPWAN	Low Power Wide Area Networks
LTE	Long-Term Evolution
M2M	Machine-to-Machine
MAC	Medium Access Control
MIC	Message Integrity Check
MVP	Minimum Viable Product

NB	Narrow Band
NLOS	Non Line of Sight
OTTA	Over The Air Activation
PAN ID	Personal Area Network ID
PER	Packet Error Rate
PHY layer	Physical layer
QoS	Quality of Service
REST API	REpresentational State Transfer API
RSSI	Received Signal Strength Indicator
SF	Spreading Factor
SNR	Signal to Noise Ratio
SRD	Short Range Device
TTN	The Thing Network
USB	Universal Serial Bus
WiFi	Wireless Fidelity

PROJECT CO-ORDINATOR CONTACT

Dr. Abdur Rahim
CREATE-NET
Via alla Cascata 56/D
Povo- 38123 Trento, Italy
Tel: (+39) 0461 408400
Fax: (+39) 0461421157
Email: abdur.rahim@create-net.org

ACKNOWLEDGEMENT

This document has been produced in the context of the H2020 WAZIUP project. The WAZIUP project consortium would like to acknowledge that the research leading to these results has received funding from the European Union's H2020 Research and Innovation Program under the Grant Agreement H2020-ICT-687607.