Investigating and Experimenting CSMA Channel Access Mechanisms for LoRa IoT Networks

Congduc Pham
Univ. Pau, LIUPPA Laboratory
congduc.pham@univ-pau.fr

***Abstract*—Given the incredible worldwide uptake of Long-Range (LoRa) networks for a large variety of innovative Internet-of-Things (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 article, 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 Carrier Sense Multiple Access (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.**

***Index Terms*—LPWAN; CSMA; Low-power IoT; Low-cost IoT; rural applications; smart villages**

I. Introduction

Recent Low-Power Wide Area Networks (LPWAN) for Internet-of-Things (IoT) introduced by Sigfox and Semtech's (LoRa™) are currently gaining incredible interest and are under intense deployment campaigns worldwide. When considering both cost and network availability constraints, LoRa technology [1], which can be privately deployed in a given area without any service subscription, has a clear advantage over Sigfox which coverage is entirely operator-managed.

While the LoRaWAN specifications [2] may ease the deployment of LoRa networks by proposing some mitigation mechanisms to allow for several LoRa networks and thousands of nodes to coexist (such as multiple channels, orthogonal spreading factors, dynamic channel discrimination) a LoRa network working in a given set of parameters still remains similar to a simple ALOHA system, which performance limitations are well-known [3]. Moreover, due to the extremely low throughput of these long-range technologies (100bps-30kbps), the time-on-air (ToA) of message can be very large, typically in the range of several seconds, thus dramatically increasing the probability of collisions despite the limitation on the duty-cycle imposed by regulations. Figure 1 shows for various combinations of bandwidth (BW) and spreading factor (SF) the ToA of a LoRa packet (obtained from formula provided in [1]) as a function of the payload size in bytes. The maximum throughput is shown in the last column with a 255B-payload packet. Modes 4 to 6 provide quite interesting trade-offs for longer range, higher data rate and immunity to interferences but in practice, when maximum range is needed, mode 1 will be the de facto standard (these are actually the default parameters in LoRaWAN). In a recent article [4], the authors have studied the scalability of LoRa networks and they confirmed the low Data Extraction Rate when the number of nodes increases.

			time on air in second for payload size of						
LoRa mode	BW (kHz)	SF	5 bytes	55 bytes	105 bytes	155 Bytes	205 Bytes	255 Bytes	max thoughput in bps
1	125	12	0.9585	2.5969	4.2353	5.8737	7.5121	9.1505	223
2	250	12	0.4792	1.2165	1.8719	2.5272	3.2645	3.9199	520
3	125	10	0.2806	0.6902	1.0998	1.5094	1.919	2.3286	876
4	500	12	0.2396	0.6083	0.9359	1.2636	1.6323	1.9599	1041
5	250	10	0.1403	0.3451	0.5499	0.7547	0.9595	1.1643	1752
6	500	11	0.1198	0.3041	0.5089	0.6932	0.8776	1.0619	1921
7	250	9	0.0701	0.1828	0.2954	0.4081	0.5207	0.6333	3221
8	500	9	0.0351	0.0914	0.1477	0.204	0.2604	0.3167	6442
9	500	8	0.0175	0.0508	0.0815	0.1148	0.1455	0.1788	11408
10	500	7	0.0088	0.028	0.0459	0.0638	0.083	0.1009	20212

Fig. 1. Time on air for various LoRa modes as payload size is varied

Given the incredible worldwide uptake of LoRa networks for a large variety of innovative IoT applications (such as image sensors) 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. Surprisingly, and to the best of our knowledge, there is limited published works discussing channel access methods for LoRa. There are mostly contributions on limitations of current LoRa technology [4], [5], [6] rather than on proposing enhancements. In this article, we investigate how a Carrier Sense (CS) mechanism can be adapted to decrease collisions in LoRa transmissions and show experimental results using our low-cost IoT LoRa testbed.

The rest of the article is organized as follows. Section II presents our low-cost & long-range IoT platform and the test-bed used for all the experiments. In Section III we will present the main Carrier Sense Multiple Access (CSMA) methods found in wireless networks such as IEEE 802.11 (WiFi) and IEEE 802.15.4. Then, in Section IV, we present the steps leading to a CSMA mechanism adapted to the specific case of LoRa technology and capable of handling both short and long LoRa messages in real-world deployment scenarios. Results and discussion will be presented. We conclude in Section V.

II. Low-cost & long-range IoT test-bed platform

Our IoT platform is developed in the context of the EU H2020 WAZIUP project. It fully takes the Arduino philosophy for low-cost, simple-to-program yet efficient hardware

platforms that is ideally well-suited for do-it-yourself (DIY) IoT, especially in WAZIUP that addresses rural applications in developing countries [8]. The Arduino-compatible ecosystem is large and proposes various board models, from powerful prototyping boards to smaller and less energy-consuming boards for final integration purposes.

The test-bed used for all the experiments presented in this article consists of several nodes and one gateway built with our low-cost & long-range IoT platform. Several types of applications, generating different LoRa message sizes, will be considered: (i) small messages, typically under 15 bytes, for simple single-sensor devices such as temperature sensors; (ii) medium-size messages, between 15 and 60 bytes, for simple multi-sensor devices (air temperature, air humidity, water temperature, dissolved oxygen level,...); and (iii) long messages, typically above 100 bytes, for advanced and innovative sensors such as image sensors.

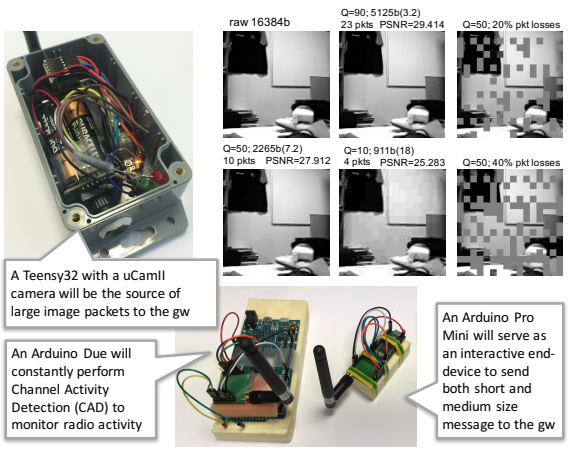

Fig. 2. Various devices of the test-bed

Fig. 2 shows the 3 devices of the test-bed. First, we have an image sensor based on a Teensy32 board and a 4D System uCamII camera configured for 8bpp gray-scale and 128x128 images that will generate long LoRa messages. The image sensor runs on 4 AA batteries and is fully autonomous with low-power features. The image encoding scheme is adapted for low-resource devices, supports high packet-loss rates and features an image quality factor parameter to ajust the compression ratio. The control software periodically takes a snapshot (one per hour for instance) and transmit the encoded image to the gateway (which will decode the image and make it available through an embedded web page). As can be seen in Fig. 2, using a quality factor of 10 offers a high trade-off between image size (compression ratio of 18) and visual quality. A typical generated image with a quality factor of 10 is about 900 bytes that can be transmitted within the ETSI limit of 36s of radio time allowed per hour. More detail on our long-range image sensor device for situation-awareness scenarios can be found in [9].

Second, we use an Arduino Due to constantly monitor the channel activity. It uses the dedicated Channel Activity Detection (CAD) features of LoRa radio chip. As CAD is an important component used for performing CS we will present this feature in more details in Section IV. This device will be attached to a computer to get the output corresponding to the observed channel activity which will be presented in graphs for better visualization.

Third, an Arduino Pro Mini running an interactive sender program can perform several tasks such as sending periodic messages of a given size as well as interactively send messages of arbitrary size. This device will be attached to a computer to get the output corresponding to the various steps of the CSMA mechanisms that will be implemented and that will be described in Section IV.

III. Channel access in wireless networks

We focus in 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 relatively 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 paper to go through all these contributions but interested readers can start with [10], [11], [12].

A. 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 random access protocols with so-called backoff procedure. Fig. 3 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
- 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 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.

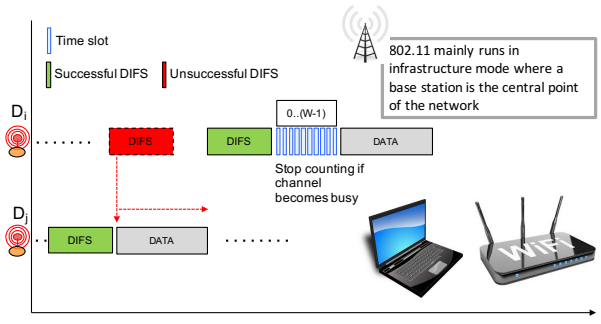

Fig. 3. IEEE 802.11 DCF CSMA/CA

B. 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 shown in Fig. 4) as the beacon-enabled needs a coordinator and higher level of synchronization that is definitely not suited for LoRa IoT networks.

- 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

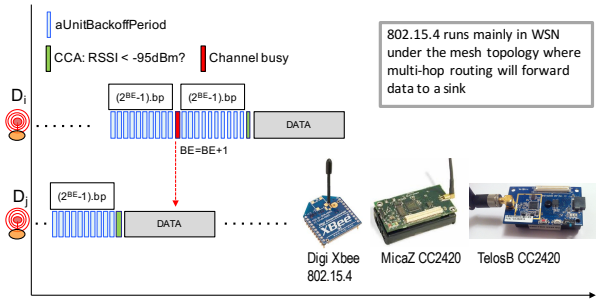

Fig. 4. IEEE 802.15.4 non-beacon unslotted CSMA

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 because 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. Reader can refer to [13] for a survey of MAC protocols for WSN.

IV. What can be done for LoRa?

A. 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 assessement, 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 Fig. 2). Fig. 5 shows 2 cases: (i) 44 byte message (40 bytes payload + 4 byte header) every 15s with a CAD procedure every 100ms and (ii) 244 byte message (240+4) every 15s with a CAD procedure every 1000ms. In Fig. 5 the red rectangle and green rectangle denote channel active duration and inactive duration respectively, and a blue spot denotes a successful CAD. As can be seen the LoRa CAD procedure can correctly detect all the LoRa transmission, and not only the preamble.

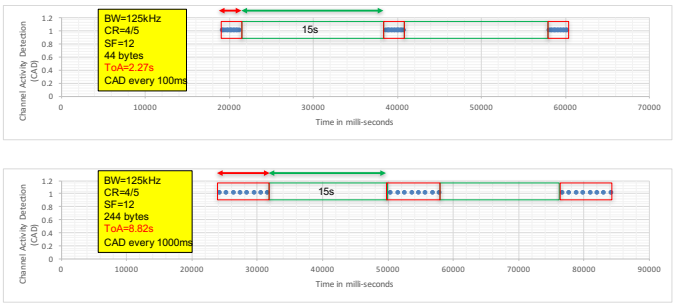

Fig. 5. Test of the LoRa CAD mechanism

B. 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 [14], it is reported that the CAD duration is between $1.75T_{sym}$ and $2.25T_{sym}$ depending on the spreading factor, see Fig. 6. We performed some experimental tests to verify the real CAD duration against what is given in [14]: Fig. 6 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

Fig. 6. 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 than 0. The $DIFS$ procedure shown in Fig. 7 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 Fig. 7, $DIFS$ is assigned 9 CAD which gives a duration of about $9 \times 61ms = 549ms$ 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 froze 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 $n \times DIFS$. For instance, if $n = 2$ then $W = 2 \times 9 = 18$ CAD.

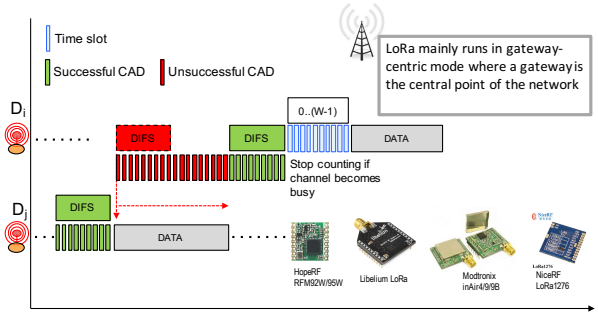

Fig. 7. CSMA mechanism adapted from IEEE 802.11

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.

Fig. 8 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.

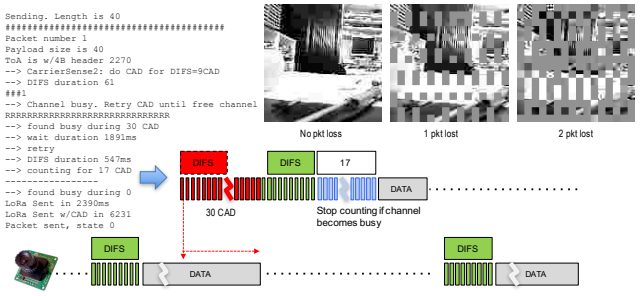

Fig. 8. Experimental test of the proposed CSMA adaptation

Fig. 8 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.

C. 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). Fig. 9 shows CAD reliability with the same traffic pattern previously shown in Fig. 5 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.

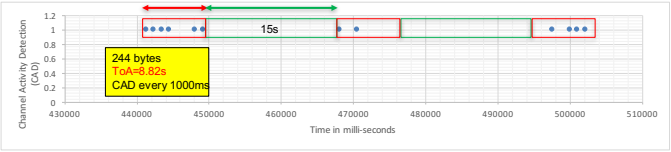

Fig. 9. CAD fails to detect activity of on-going transmissions

This CAD unreliability issue in real-world deployment scenario has a huge negative impact on the CS mechanism. For instance, in the previous 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 Fig. 5 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 have always been some successful CAD during any transmission.

D. 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 (see Fig. 1). During this extended $DIFS(ToA_{max})$, CAD procedure is performed periodically (for instance every 1000ms as in Fig. 5–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.

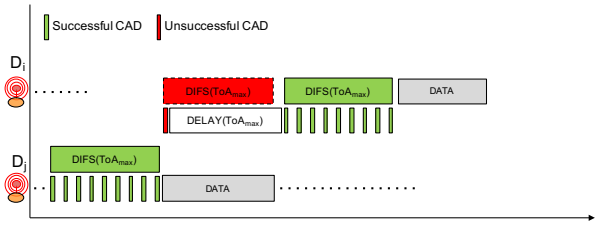

Fig. 10. New CSMA proposition

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

E. Discussions

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 Fig. 6 while the ToA of the longest LoRa packet, ToA_{max}, is respectively between 100ms and 9150ms as shown in Fig. 1. 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 IV-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.

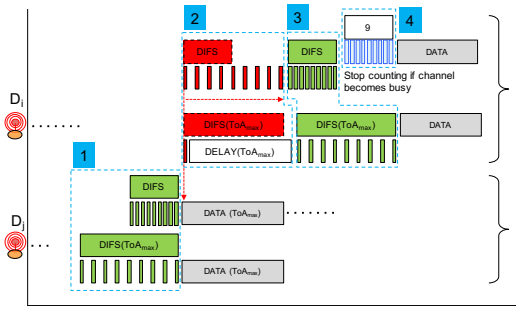

Fig. 11. Scenario for comparing $CSMA_{802.11}^{LoRa}$ and $CSMA_{new}^{LoRa}$

2) Energy considerations: We can compare the energy consumption between $CSMA_{802.11}^{LoRa}$ and $CSMA_{new}^{LoRa}$ with the scenario depicted in Fig. 11: 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 Fig. 11 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 $E_{DIFS} = E_{DIFS(ToA_{max})} = 9 \times E_{CAD}$. 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 $9 \times E_{CAD}$. 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 $1 \times E_{CAD}$.

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 Fig. 11, $CSMA_{802.11}^{LoRa}$ 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 block 2 for $CSMA_{new}^{LoRa}$ will have an energy consumption of $2 \times [9 \times E_{CAD}] + N \times \frac{1}{P_{CAD}} \times E_{CAD}$. Fig. 12 shows the energy consumption when varying N and P_{CAD}.

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 $CSMA_{new}^{LoRa}$.

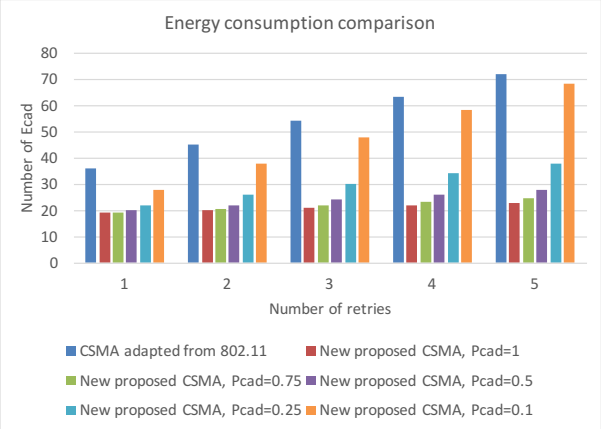

Fig. 12. Energy comparison of $CSMA_{802.11}^{LoRa}$ and $CSMA_{new}^{LoRa}$

V. Conclusions

In this article, 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 others.

Acknowledgments

This work is supported by the WAZIUP project with funding from the EU's Horizon 2020 research and innovation programme under grant agreement No 687607.

References

[1] Semtech, "LoRa modulation basics. rev.2-05/2015," 2015.
[2] L. Alliance, "LoRaWAN specification, v1.0.2," 2016.
[3] R. Nelson and L. Kleinrock, "The spatial capacity of a slotted ALOHA multi-hop packet radio network with capture," *IEEE Trans. Comm.*, vol. 32, 1984.
[4] M. C. Bor, U. Roedig, T. Voigt, and J. M. Alonso, "Do LoRa low-power wide-area networks scale?" in *ACM MSWiM'16*.
[5] D. Bankov, E. Khorov, and A. Lyakhov, "On the limits of LoRaWAN channel access," in *EnT'16*.
[6] K. Mikhaylov, J. Petaejaejaervi, and T. Haenninen, "Analysis of capacity and scalability of the LoRa low power wide area network technology," in *22th European Wireless Conference*, 2016.
[7] ETSI, "Electromagnetic compatibility and radio spectrum matters (ERM); short range devices (SRD); radio equipment to be used in the 25MHz to 1000MHz frequency range with power levels ranging up to 500mW; part 1: Technical characteristics and test methods," 2012.
[8] C. Pham, A. Rahim, and P. Cousin, "Low-cost, long-range open IoT for smarter rural african villages," in *IEEE ISC2'16*.
[9] C. Pham, "Low-cost, low-power and long-range image sensor for visual surveillance," in *ACM SMARTOBJECTS'16*, 2016.
[10] M. Kaynia and N. Jindal, "Performance of ALOHA and CSMA in spatially distributed wireless networks," in *IEEE ICC'08*.
[11] Y. Yang and T.-S. P. Yum, "Delay distributions of slotted ALOHA and CSMA," *IEEE Trans. Comm.*, vol. 51, 2003.
[12] F. A. Tobagi, "Distribution of packet delay and interdeparture time in slotted ALOHA and CSMA," *J. Assoc. Comput. Mach.*, vol. 29, 1982.
[13] A. Bachir, M. Dohler, T. Watteyne, and K. K. Leung, "MAC essentials for WSN," *IEEE Comm. Surveys and Tutorials*, 12(2), 2010.
[14] Semtech, "Sx1272/73 - 860MHz to 1020MHz low power long range transceiver. rev.2-07/2014," 2014.