Decentralized Spectrum Learning for Radio Collision Mitigation in Ultra-Dense IoT Networks: LoRaWAN Case Study and Experiments

Christophe Moy[1], *Senior Member, IEEE*, Lilian Besson[2], Guillaume Delbarre[1], Laurent Toutain[3]

[1] Univ Rennes, CNRS, IETR - UMR 6164, F-35000, Rennes, France

[2] CentraleSupélec, CNRS, IETR - UMR 6164, F-35576, Cesson-Sévigné, France

[3] IMT Atlantique, IRISA, F-35700, Rennes, France

Corresponding author e-mail : christophe.moy@univ-rennes1.fr

***Abstract*—This paper describes the theoretical principles and experimental results of reinforcement learning algorithms embedded on IoT devices, in order to tackle the probem of radio collision mitigation in ISM unlicensed bands. Multi-Armed Bandit (MAB) learning is here used to improve both the IoT network capability to support the expected massive number of objects, as well as the autonomy of the IoT devices. We first illustrate the efficiency of the proposed approach in a proof-of-concept, based on USRP software radio platforms operating on real radio signals. It shows how collisions with other RF signals are diminished for a given IoT device that uses MAB learning. Then we describe the first implementation of such algorithms on LoRa devices operating in a real LoRaWAN network, that we named IoTligent. The proposed solution adds neither processing overhead, so it can be ran in the IoT devices, nor network overhead, so no change is required to LoRaWAN protocol. Real life experiments done in a real LoRa network show that IoTligent devices' battery life can be extended by a factor 2, in the scenarios we faced during our experiment. Finally we submit IoTligent devices to very constrained conditions that are expected in the future with the growing number of IoT devices, by generating an artificial IoT massive traffic in anechoic chamber. We show that IoTligent devices can cope with spectrum scarcity that will occur at that time in unlicensed bands.**

Keywords—Internet of Things, IoT, machine learning, MAB, bandit, radio spectrum, collision mitigation, interference, LoRa, artificial intelligence, LoRaWAN, cognitive radio.

I. INTRODUCTION

Wireless Internet of Things (IoT) is based on Low Power Wide Area Networks (LPWAN) able to interconnect low cost and mostly battery-powered devices over long ranges to an access point to the Internet. This is made possible by the use of low bit rates, low-bandwidth machine-to-machine (M2M) types communications. After the expansion of human-to-human mobile communications in the 1990's, and then human to the Internet communications in the 2000's, now has come the era of M2M and especially Machine to the Internet (M2I). M2I are expected to know a tremendous expansion in the very next few years, through IoT networks.

We can consider two categories of IoT networks. First are the cellular IoT networks, deployed by mobile phone operators, running 3GPP standards such as EC-GSM IoT, LTE-Cat0, LTE-Cat M1, NB-IoT or next 5G IoT. These standards will be supported in licensed frequency bands operated for cellular telephony. The second category is mainly different as it uses unlicensed bands for wireless links, also called ISM bands, which are open to the use for Industrial Scientific and Medical applications. Most commonly used ISM bands are 434 MHz and 868 MHz in Europe and Africa, and 915 MHz in America, with a worldwide bands at 2.4 GHz and 5.8 GHz. Due to the constraints in terms of range and bandwidth, the 868 MHz and 915 MHz bands are mostly preferred for IoT networks. They communicate through protocols based on very different radio physical layer and medium access control specifications. For instance, the current two most well known IoT standards are LoRaWAN [1], based on a chirp spread-spectrum solution and Sigfox [2], based on an ultra-narrow band technology.

In cellular licensed IoT networks, just one transmission may occur between any operated device and the radio access point, scheduled by the cellular network in a given place, at a given time and in a given frequency band. However, in unlicensed bands, IoT networks face very different and specific conditions. Many IoT networks can be deployed in the same area and superpose geographically, regardless if they are using the same protocol or not. Even if there exist rules to be followed in unlicensed bands, such as transmit power mask and duty cycle limits, many radio transmissions may collide at the same place, time and frequency, as no global coordination is done.

The goal of this paper is to present the original IoTligent approach that embeds very low cost machine learning algorithms inside IoT devices, in order to mitigate radio collisions in the ISM bands. Low cost here is to be considered in terms of processing power, processing resources, memory footprint, protocol overhead and frequency resources usage.

After exposing the issues we target in this work and the corresponding hypothesis in Section II, Section III reminds the foundation of the learning algorithms used in IoTligent. Then, we show how we validated our approach through several gradual stages of experimentations. The first measurements of Section IV give results of a proof-of-concept made in laboratory conditions using SDR (Software Defined Radio) platforms in order to validate the learning approach. Then, Section V gives the experimental architecture and hardware configuration we use for the next measurement campaign presented in Section VI. It has been realized on LoRa IoT devices operated in real radio conditions of an operating LoRaWAN network in the city of Rennes (France). In Section VII, we present measurements made in an anechoic chamber

with an emulated radio traffic generator. We reproduce here the future very dense IoT networks radio conditions and validate the proposed learning approach for future ultra-dense LoRaWAN networks.

II. COLLISIONS, HYPOTHESIS AND ADVANTAGES OF DECENTRALIZATION

A. Collisions vs. autonomy

The possibility of suffering from collisions is the main drawback of IoT in terms of battery autonomy at the first level, but also of IoT viability itself in the ISM bands. Indeed collisions may cause (many) retransmissions at the cost of an increase of the RF contention, and may lead to a lower battery lifetime of the devices. Even worse, this could lead to a total failure of the IoT device, either because it cannot succeed in sending any data to the network, or because multiple repetitions could make it consume all its energy much faster than expected.

B. Analysis of collisions

Radio collisions will be the weak point of LPWAN IoT networks operating in the unlicensed bands. Different kinds of collisions exist, as collision may occur with:

- other IoT devices of the same network, as several networks covering the same area are not coordinated. This can occur between IoT devices uplink (UL) transmissions, and between IoT UL and gateway downlink (DL) transmissions towards IoT devices.
- Other IoT devices of surrounding networks that are not the network of our device, but that are using the same IoT standard. This can occur both in UL and DL, as surrounding IoT gateways of different networks are *not* coordinated. They could use the same channels, or partly same and partly different channels.
- Other IoT radio signals using other IoT radio standards with different channels, bandwidth, user repartition, *etc.*
- Other radio signals present in the ISM bands that are not IoT signals. By definition, they use completely different rules than IoT. They can be considered as "jammers" from the IoT network point of view.

It is also important to note that, as each IoT standard uses its own rules for channeling and bandwidth, all this leads to an erratic spectrum usage, which cannot be planned, and has to be learnt *in vivo*. However, unlicensed band does not mean un-ruled band (there are for duty cycle, power, *etc.*), but they are more exposed to the non-respect of these few rules as regulation is relaxed and thus, controls as well.

C. A device-side solution for spectrum management

Our learning approach imposes no change on a normal IoT protocol, as for instance LoRaWAN [1]. It means that there are no extra-retransmission, no data to be added in frames, no extra-power transmission, *etc.* to be done. The only condition is that the proposed solution should work with the *acknowledged* mode for IoT. The underlying hypothesis is that "channels" (there are no official channels in ISM bands) occupancy by surrounding radio signals (IoT or not) is not equally balanced. In other words, some ISM sub-bands are less occupied or less jammed than others, but it is not possible to predict it in time and space, so it has to be learnt on the fly.

The considered learning algorithms are a kind of artificial intelligence (AI) algorithms that are compatible with the constraint of low complexity of IoT devices, as we explain below. It is indeed much more efficient to implement a radio collision mitigation approach on the device side, as devices may be quite far away from gateways, and suffer from different radio and jamming/co-existence conditions. But they are the place where every Watt counts at transmission, and where sensitivity should be the best at reception, as no extra-processing can be afforded.

D. Advantages of the proposed solution

The proposed approach is based on reinforcement learning algorithms such as those already studied [3] and experimented on real radio signals for Cognitive Radio, and especially for Opportunistic Spectrum Access (OSA) [4]. We assert that, as for OSA, the IoT spectrum access issue can be modeled as a Multi-Armed Bandit (MAB) problem [5][6]. Reinforcement Learning is based on a feedback loop that gives a success/failure measure of experience. In the IoT context, we propose to use the acknowledgement (ACK) sent by the gateway to the IoT device as a binary reward (1/0 for presence/absence of ACK). Every device aims at maximizing its transmission success rate, or equivalently, at maximizing its cumulated reward (i.e. number of received ACK).

The main advantages of our solution are that the

- algorithms have mathematical proofs of convergence,
- proofs are verified in real radio conditions, thanks to the good matching between the model and reality,
- learning converges effectively very fast in real experiments, thus it is adequate for radio applications [7],
- implementation and execution both require very low processing and memory overhead, so that it is possible to add the proposed approach in IoT devices for a negligible money cost, negligible complexity (processing, hardware, memory) and negligible extra-energy consumption overhead,
- learning can efficiently start from scratch, so there is no need for any prior knowledge when deploying the IoT device (i.e. no need to loose some time to acquire this knowledge before operation really starts),
- using such learning algorithms will never give worse results than a state-of-the-art random solution [8], even before learning brings a clear advantage, for instance at the very beginning of the learning process.

Hence, we argue that the proposed approach can adapt to any kind of radio context, and we also note that:

- the stationarity of the environment is a requirement for the proofs of convergence, but if conditions change occasionally, convergence is so fast that a simple solution consists in resetting learning from time to time [8] (note that there also exist adaptive versions),
- no coordination is required between devices, but benefits decrease with the number of devices using the proposed solution, when it represents a great majority of devices (see the solutions presented in [8][9]),
- as soon as a device is planned to receive an acknowledgment, no overhead is added neither in terms of protocol nor extra bits to be put into the LoRaWAN frames in uplink or downlink. A received ACK yields a reward of 1, and no ACK yields a reward of 0, without needing to change the content of the ACK messages.

III. MAB MODEL AND LEARNING SOLUTIONS

We model the IoT wireless spectrum issue as a Multi-Armed Bandit (MAB) problem [5] and we propose to use bandit algorithms at the IoT device side to solve this issue [6].

A. System model

We consider the system model presented in Fig. 1, where a set of devices sends uplink packets to the network gateway.

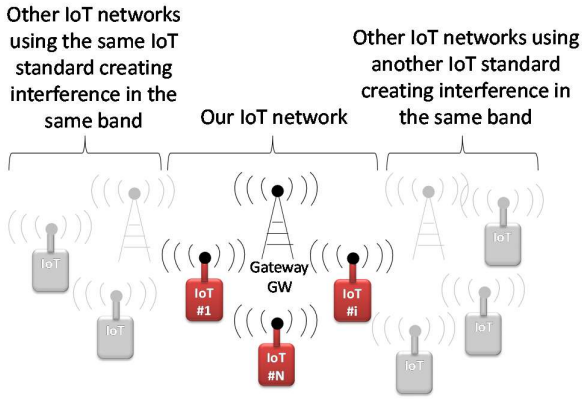

Fig. 1. System model used for IoT, with intelligent IoT devices that are able to dynamically set their transmission channel, thanks to a learning algorithm, in order to minimize collisions and interference from other radio signals in the unlicensed ISM band, especially other IoT networks which will be responsible of most of future traffic.

The communications between IoT devices and this gateway are done through a simple ALOHA-based protocol, where devices transmit uplink packets of fixed duration, whenever they want. The devices can transmit their packets in one of the $K \geq 2$ channels. Channels are predefined but time is unslotted. In the case where the gateway receives an uplink in one channel, it transmits an acknowledgement to the corresponding end-device in the same channel, after a fixed delay. These communications operate in unlicensed ISM bands, and consequently, as stated in the previous section, they suffer in particular from interferences generated by uncoordinated neighboring networks. This interfering traffic is uncontrolled, and can be unevenly distributed over the K different channels.

We consider the network from the point of view of a single IoT device. Every times it has to communicate with the gateway (at each transmission $t \geq 1$, $t \in \mathbb{N}$), it has to choose one channel, denoted as $C(t) = k \in \{1, \dots, K\}$. After transmission, the IoT device starts to wait in the same channel $C(t)$ for an acknowledgement sent by the gateway. Before sending another message (i.e., at time $t + 1$), the IoT device knows if it received or not this ACK message. For this reason, selecting the channel (or *arm*) k at time t yields a (random) feedback, called a *reward*, $r_k(t) \in \{0, 1\}$, being 0 if no ACK was received after the previous message, or 1 if ACK was successfully received. The goal of the IoT device is to minimize its packet loss ratio, or equivalently, it is to maximize its successful transmission rate, which here is its cumulative reward, as it is usually done in MAB problems [5][6][10]:

$$r_{1\dots T} := \sum_{t=1}^{T} r_{C(t)}(t) \qquad (1)$$

This problem is a special case of the so-called "stochastic" MAB, where the sequence of rewards drawn from a given arm k is assumed to be *i.i.d.*, under some distribution ν_k, that has a mean μ_k. Several types of reward distributions have been considered in the literature, for example distributions that belong to a one-dimensional exponential family (e.g., Gaussian, Exponential, Poisson or Bernoulli distributions). As rewards are binary in our model, we consider only Bernoulli distributions, in which $r_k(t) \sim \text{Bern}(\mu_k)$, that is, $r_k(t) \in \{0, 1\}$ and $\mathbb{P}(r_k(t) = 1) = \mu_k \in [0, 1]$. Contrary to many previous works done in the cognitive radio field (for instance in Opportunistic Spectrum Access [11]), the reward $r_k(t)$ does *not* come from a sensing phase before sending the t-th message, as it would do for any "listen-before-talk" model. Rewards come from receiving an acknowledgement from the gateway, between the t-th and t+1-th messages.

The problem parameters $\mu_1, \dots, \mu_K$ are of course unknown to the IoT device, so to maximize its cumulated reward, it must learn the distributions of the channels, in order to be able to progressively focus on the best arm (i.e., the arm with largest mean). This requires to tackle the so-called exploration-exploitation dilemma: a player (here, an IoT device) has to try all arms a sufficient number of times to get a robust estimate of their qualities, while not selecting the worst arms too much.

Before discussing the relevance of a MAB model for our IoT application, we present two low-complexity bandit algorithms, UCB_1 and Thompson Sampling [12], which are both known to be efficient for stationary *i.i.d.* rewards and are shown below.

B. The UCB_1 algorithm

A first naive approach could be to use an empirical mean estimator of the rewards for each of the K channels, and select the channel with the highest estimated mean at each time ; but this "greedy" approach is known to fail dramatically [5]. Indeed, with this policy, the selection of arms is highly dependent on the first draws: if the first transmission in one channel fails and the first one on other channels succeed, the device will never use the first channel again, even it is the best one, which is the most available one, in average.

Upper Confidence Bounds (UCB) algorithms instead use a confidence interval on the unknown mean μ_k of each arm, which can be viewed as adding a "bonus" exploration to the empirical mean. They follow the "optimism-in-face-of-uncertainty" principle: at each step, they play according to the best model, by selecting the statistically best possible arm (i.e., the highest upper confidence bound). More formally, for one IoT device, we denote by

$$T_k(t) = \sum_{\tau=1}^{t} \mathbb{1}(C(\tau) = k) \qquad (2)$$

the number of times channel k was selected up-to time $t \geq 1$. The empirical mean estimator of channel k is defined as the mean reward obtained by selecting it up to time t,

$$X_k(t) = (1/T_k(t)) \sum_{\tau=1}^{t} r_k(\tau) \mathbb{1}(C(\tau) = k) \qquad (3)$$

For UCB_1 [6],the confidence term is

$$A_k(t) = \sqrt{\alpha \log(t)/T_k(t)}, \qquad (4)$$

and the upper confidence bound is the sum of the confidence term and the empirical mean,

$$B_k(t) = X_k(t) + A_k(t), \qquad (5)$$

which is used by the device to decide the channel for communicating at time step $t + 1$:

$$C(t+1) = \underset{1 \le k \le K}{\arg\max} B_k(t) \quad (6)$$

The UCB_1 algorithm is called an index policy. It uses a parameter $\alpha > 0$, originally set to 2 [13], but empirically $\alpha = 1/2$ is known to work better (uniformly across problems), and $\alpha \geq 1/2$ is advised by the theory [13]. This algorithm is simple to implement and to use in practice, even on embedded microprocessors with limited computation and memory capabilities. In our model, every IoT device implements its own UCB_1 algorithm, independently. For one IoT device, the time t is the total number of sent messages from the beginning, as rewards are only obtained after a transmission. Different devices do not share this time index t as time is not slotted.

C. The Thompson Sampling algorithm

Thompson Sampling (TS) [12] was introduced early on, in 1933 as the very first bandit algorithm, in the context of clinical trials (in which each arm models the efficacy of one treatment across patients). Given a prior distribution on the (unknown) mean of each arm, the algorithm selects the next arm to draw based on samples from the conjugated posterior distribution, which for Bernoulli rewards is a Beta distribution.

A Beta prior Beta($a_k(0) = 1$, $b_k(0) = 1$) (initially uniform) is assumed on $\mu_k \in [0, 1]$, and at time t the posterior is denoted Beta($a_k(t)$, $b_k(t)$). After every channel selection, the posterior is updated to have a_k(t) and $b_k(t)$ counting the number of successful and failed transmissions made on channel k. More precisely, if the ACK message is received, the update is $a_k(t+1) = a_k(t) + 1$, and $b_k(t+1) = b_k(t)$, otherwise the update is $a_k(t+1) = a_k(t)$, and $b_k(t+1) = b_k(t) + 1$. Then, the decision is done by sampling an index for each arm, at each time step t, from the arm posteriors: $I_k(t) \sim$ Beta($a_k(t)$, $b_k(t)$), and the chosen channel is simply the channel $C(t+1)$ with highest index $I_k(t)$. For this reason, Thompson Sampling can be called a *randomized* index policy.

The TS algorithm, although being simple and easy to implement, is known to perform well for stochastic problems, for which it was proven to be asymptotically optimal [14][15]. It is known to be empirically efficient, and for these reasons it has been used successfully in various applications, including on problems from Cognitive Radio [16][17], and also in previous works on decentralized IoT-like networks [18].

D. Multi-player bandit issue

We can prove that one single intelligent IoT can improve consequently its performance in LPWAN IoT networks using unlicensed band. But we have also shown that even if there are a lot of intelligent IoT devices, and the model of other surrounding IoT devices does not stay purely stochastic, learning still brings improvement [8]. Further theoretical developments on this direction are an interesting future work.

IV. Measurement 1 : IoT Proof-of-Concept

A. Preceding results

Bandit algorithms have been identified more than 10 years ago as efficient solutions for many cognitive radio problems, as introduced in [3]. In particular, the very trendy dynamic spectrum access (DSA [11]) issue has been identified as a multi-armed bandit (MAB) problem in [4]. The first implementation validating the bandit algorithms on real radio signals was presented 5 years ago for opportunistic spectrum access (OSA) in [7]. Reinforcement learning algorithms, such as UCB_1, were firstly used, but any kind of bandit algorithm [19] could be used indifferently. Their efficiency and implementation complexity can be considered as criterion to decide which algorithm to implement. In the context of IoT, MALIN [20] is the first proof-of-concept (PoC) demonstrating the feasibility of using learning algorithms on the IoT device side, on real radio signals in lab conditions.

B. PoC setup

This PoC is based on 4 USRP platforms from Ettus Research and National Instrument[1]. The development is made with GNU Radio[2] software, and the source code of the PoC is published on-line[3], in order to ease the full reproducibility of our results. We have not implemented a real IoT standard in this PoC, in order to show that it can be applicable for any IoT standard. However, we took some characteristics rather corresponding to the LoRa context (not ultra-narrow band, reduced number of channels, frame duration around a few hundreds of milliseconds, *etc.*).

One or two (or more) USRP platforms are playing the role of IoT devices that can run (or not) the proposed learning algorithms. They transmit at their own initiative some very light modulated information (using QPSK), in order to be identified by the gateway, and then wait during one second for the gateway ACK. Both uplink transmissions and downlink receptions use the same frequency channel. Whether the ACK is received or not, the learning algorithm updates its knowledge about the channel used during this iteration.

One USRP platform is a traffic generator that emulates as much (random) IoT traffic as we want, to be able to tune each channel's load independently, on demand. We typically choose channel loads ranging from 0% to 20%.

A last USRP platform is a gateway (GW) that is continuously scanning all the K channels, and monitors the IoT traffic composed of the artificial signals produced by the traffic generator and the IoT devices signals. The gateway has the ability to answer to the IoT devices, by sending back to them an ACK message, which contains their identifier, (actually, the symbols corresponding to the QPSK complex conjugate of their identifier).

C. PoC results

The number of IoT channels K is a parameter, and we have set it to 4, 8 and 16 channels in our experiments, but there is no limitation. For the sake of clarity in the figures, we give examples below with 4 channels that are separated by empty channels, but they could be contiguous with no change neither in the implementation nor in the results.

We can see on Fig. 2 a time-frequency waterfall view captured by the gateway, where we can observe the RF traffic in the K=4 channels. The y axis for the time is vertical and goes down, and frequency is on the x axis. The difference of colors is a difference of received power, due to the distance of the transmitters to the gateway receiver antenna. The gateway transmitter antenna is very close so signals transmitted by the gateway are red. The traffic generator and IoT devices are a little bit further away, so the gateway received weaker signals

[1] See https://www.ettus.com/ for more details.

[2] See https://www.gnuradio.org/ for more details.

3 See https://bitbucket.org/scee_ietr/malin-multi-armed-bandit-learning-for-iot-networks-with-grc. The code is released publicly under the open-source GPLv3 license.

from them: one is blue and the other green, which reveals a low difference.

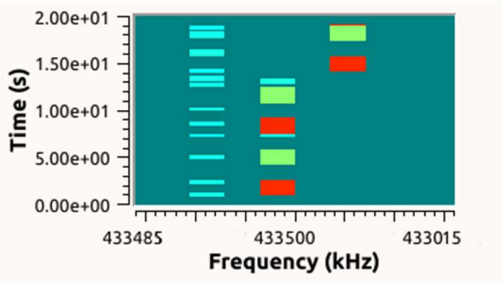

Fig. 2. Spectrum waterfall on GRC received at gateway side in a 4 channels example (only 3 occupied in this picture), during experiments. Time is in *y* axis (going down) and frequency in *x* axis. Blue short transmissions are those produced by the traffic generator, green blocks are our IoT transmissions and red blocks are the gateway transmissions itself.

In this experiment, we can see on Fig. 2 that channel #0 on the left hand side faces a dense IoT traffic, which appears as blue short transmissions (produced by the traffic generator). Some others uplink transmissions appear on channel #1 (second left hand side), but we do not see any blue short messages on channel #2 (third left) and #3 (on the right hand side, empty in this measure). However, we see on these channels longer messages of two kinds: green messages which correspond to IoT devices transmissions, and red messages that are the answer done by the gateway. In order to rapidly have results in the demo, we make them transmit every 5 seconds, for a message of duration of one second. Then when an IoT device transmits a message, the gateway should answer and sends an ACK to the IoT device within 1 second, if the gateway was able to demodulate the signal, i.e., if there is no collision in the radio channel. For instance, we can see on Fig. 2 that the IoT device moved from channel #2 to channel #1, and at each of its transmission, the gateway was able to answer, by successfully sending an ACK response.

Fig. 3 gives the perspective of the IoT device, at a different moment for the same scenario. Then we observe that colors have changed, as the received power is now taken at the device side. The IoT device transmitter antenna is now very close, so signals transmitted by the IoT device are red. The traffic generator, the other IoT devices, and then the gateway all are a little bit further away, so the IoT device received weaker signals from all of them, one is blue and the other green but inversed. However, it is not so obvious, so it is better to consider the message duration in the *y* axis indeed.

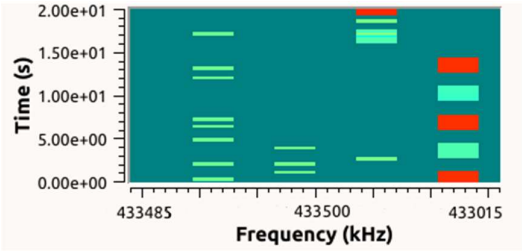

Fig. 3. Spectrum waterfall on GRC received at IoT device side in a 4 channels example, during experiments. Time is in *y* axis (going down) and frequency in *x* axis. Green short transmissions are those produced by the traffic generator, red blocks are the IoT device transmissions, and blue blocks are gateway transmissions.

We can see on Fig. 3 that if we use the same scenario of traffic as in Fig. 2, but at a different time, i.e. with a very dense traffic on channel #0, less dense on channel #1, even less dense on channel #2, then transmission appears on channel #3 but it is indeed just even less dense. At that time of the experiment, our IoT device is moving from channel #2, where maybe it faced some collisions in the downlink transmission of ACK, to channel #3, where several successive transmissions and receptions seem to occur.

Fig. 4 is a screenshot taken at some moment during an experiment that gives the details of the learning algorithm operation. We can see in top-left red data the number of selections of each channel. There is a clear disequilibrium with channel #3 that has been much more (17 times) used than channel #2 (8 times), itself more used than channel #1 (6 times) and channel #0 (only once). This reveals the effect of the learning algorithm. It has analyzed which channels are more occupied and more disturbed by other users of the band (emulated here by the traffic generator). The top-right green and therefore the bottom-right blue data explain such a choice. Channel #4 has known 16 successes (over 17), so a rate of 94%. We remind that successes mean that the IoT device received on that channel 16 ACK from the gateway after transmitting 17 times in this channel. So just one "exchange" was lost, either in UL, or in DL, due to a collision with some interfering signal in the channel. We can see on the opposite that no success has been obtained for channel #0, so it has a 0 % rate. UCB data, in bottom-left green part, are harder to follow, as UCB_1 indexes rapidly converge to very close values, but at each transmission, the IoT device chooses channel with highest UCB_1 index, as in (6).

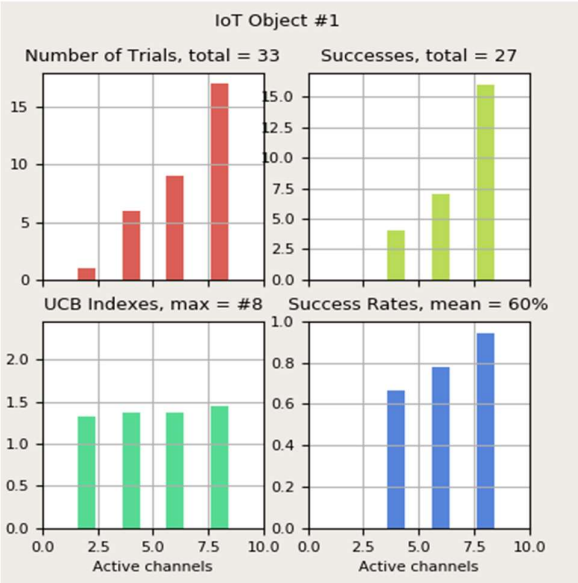

Fig. 4. Live results enabling to monitor the learning algorithm evolution at the IoT device side in a K=4 channels example. Top-left red: number of trials on each channel, top-right green: number of successes on each channel (ACK received by IoT device), bottom-left green: UCB_1 index $Bk(t)$ for each channel, bottom-right blue: success rate on each channel.

V. Experimental Architecture and Hardware Configuration for Real LoRa Measurements

The next step after the previously exposed proof-of-concept consists in implementing the same approach in real conditions of operation, that is, in a real IoT network and not only in laboratory conditions. We target here a LoRaWAN

[1][21] IoT context, but it could be done with any other IoT standard, as soon as it uses acknowledgment feedback. We describe the involved implementation details in this section.

As far as the authors know, this is the first implementation of decentralized artificial intelligence algorithms in IoT devices to tackle the IoT spectrum contention mitigation problem. We named our approach IoTligent [22][23]. It is first necessary to remind quickly how a LoRaWAN network is constituted. We are using here a real LoRa network in the European ISM band, at 868 MHz.

A. *LoRaWAN architecture*

The implementation of the learning algorithm we propose is decentralized, it takes place only on the LoRa device side. As stated earlier, it impacts no aspect of the LoRaWAN network. We explain below a little bit more the LoRaWAN network side configuration, and we refer to [21] for more details. LoRaWAN network, as any other IoT network, can be summarized by four main elements, as shown in Fig. 5:

- LoRa devices (our devices run the UCB_1 algorithm here),
- one or more LoRa gateway(s) receiving all LoRa radio signals in their radio range,
- a Lora Network Server (LNS) that discriminates devices subscribing to its network from others,
- an Application Server (AS) that receives the data sent by devices and sends back ACK to them (mandatory here).

The IoT devices are associated to a given LoRaWAN network with a "join phase", at their very first communication through a gateway of this network. The LNS is in charge of the association, as explained below. Finally, data extracted from radio signals, sent by the IoT devices, are sent to the Application Server (AS) that manages data (i.e. processes them, sends them to a storing place in the cloud and/or an application). Then the role of the AS is to initiate a sending of an ACK to the IoT device, through LNS and a gateway, down to the IoT device.

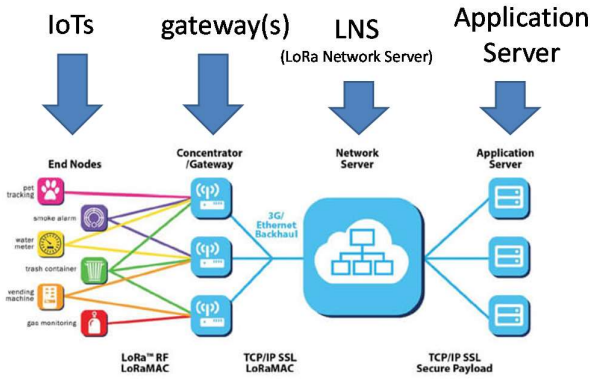

Fig. 5. LoRaWAN network parts: IoT devices, gateways, LNS and AS [21].

B. *Device side*

For this experiment, we implement an IoT device by using a Pycom card[4] composed of an Expansion Board and a LoPy 4 module which can support LoRa wireless connectivity, as shown on Fig. 6. The Pycom card is programmed in the Python language. The frequency channels used in the experiments are those authorized in France, the country of experimentation. Our IoTligent proposal is agnostic to *K*, the number of channels in the standard, and thus it can be used in any country.

We had to make some modifications in the LoRa library written in C and the ESP32 chip library written in MicroPython. By default, the Pycom configuration for Europe is to use only 3 channels in a random manner : 868.1, 868.3 and 868.5 MHz (with a duty cycle of 1%). So, for measurement 3 of Section VII, we added a custom configuration region in the LoRa library with 16 channels, covering the band from 865.9 to 868.9 MHz. We added the possibility for ESP32 chip to force a channel in *LORAWAN* mode, what is necessary in order to follow UCB_1 policy for both measurement 2 and 3 of the two following sections.

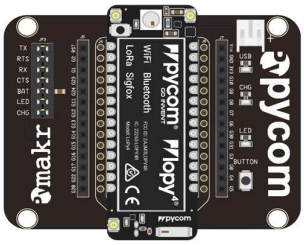

Fig. 6. Pycom module composed of a LoPy4 and an Expansion Board.

C. *LoRa gateways*

For measurement 2 of the next Section VI, we use outdoor LoRa gateways operated by Acklio Company that has several gateways deployed in the city of Rennes, where the experiments were made. We did not have access to their configuration, so only the 3 default channels have been used for this measurement campaign.

But for measurement 3 of Section VII, we use our own indoor gateway shown in Fig. 7, whose channel parameters could be changed in order to adapt the number of channels depending on our measurement needs. This is done by changing the configuration file of the Semtech SX1301 chip which manages two radio SX1257 chips. It consists in chosing the central frequency of the two radio chips and choose the offset in an interval of ±500k Hz, for each channel.

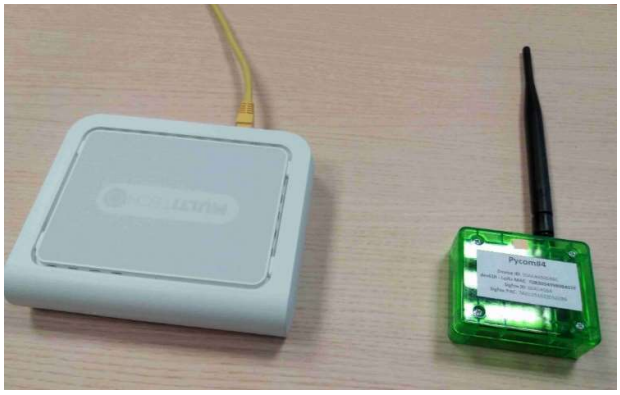

Fig. 7. Indoor gateway used for Measurement 3 experiments on the left side, and packaged Pycom device on the right side.

[4] Pycom documentation: https://GitHub.com/PyCom/PyCom-libraries

D. Network side

We have access to the LNS provided by Acklio Company. The LNS sends the received messages to an AS which is a Linux server, running in the cloud. The AS is running a Python program that enables to display data and metadata (i.e., frequency, time of reception, *etc*). This programs also contains instructions to send an acknowledgment to the device, using in DL the same frequency used by the IoT device at UL.

VI. Measurement 2 : IoTligent Operation in a Real LoRaWAN Network

A. Device side configuration

We use the *LORAWAN* mode with an Over-The-Air-Activation (OTAA) using *app_EUI* and *app_key* keys, as shown in the following Python code for the Pycom device:

```
lora = LoRa(mode=LoRa.LORAWAN, region=LoRa.EU868)

lora.join(activation=LoRa.OTAA,    auth=(app_EUI,
app_key), timeout=0)
```

The transmit channel frequency is then chosen in a set of K channels, which is set here at K=3 in this experiment. We use standard Europe UL channels with the following frequency table (in Hz):

```
tabFreq =[868100000, 868300000, 868500000]
```

The IoTligent device infinite *while loop* is started, running the algorithm presented in the previous section and [5], in order to choose which frequency to be selected at each iteration before executing a send operation. An ACK is then expected from the network side in *non-blocking* mode so that when ACK is not received, the device just updates its learning data and still goes on.

B. Network side – Lora Network Server (LNS)

The different IoTligent devices should be declared to the LNS, with at least the following information:

- *devEUI* : ID of the device obtained by executing a « *get_id.py* » program[4] on the Pycom device itself,
- *appEUI* : which should correspond to *app_eui* chosen in the Pycom device,
- *appKey*: which should correspond to *app_key* chosen in the Pycom device,
- other parameters are let by default at SF=12 (spreading factor), and bandwidth BW=125 kHz.

The address of the AS is also specified in *Connectors*, as well as the mode used to send data between LNS and AS (we chose *http callback* here).

C. Network side – Application Server (AS)

The AS runs a Python program that receives data from the LNS, as well as LoRa metadata with all parameters of the LoRaWAN transmission (frequency, SF, BW, time of arrival, *etc*). This program also sends an acknowledgment message to the device in DL. First, an acknowledgment attempt is sent by default at the same frequency than the message transmitted by the device it answers to. Then we block any other retransmission. This is exactly what is necessary for the learning process of IoTligent:

- to use the same channel in both UL and DL,
- to avoid retransmission in order to increase the battery durations of devices on the one hand, and radio frequency overload on the other hand.

D. Learning algorithm in Pycom device

The learning algorithms used in IoTligent are (any) bandit algorithms, such as those first used for Cognitive Radio dynamic spectrum access in [4], and implemented in the exhaustive open-source SMPyBandits Python library [19]. We take here the example of UCB_1 algorithm, as presented above. We have chosen this algorithm as it is known to be efficient and to converge quickly, and also for its ease of implementation. The only data necessary to be stored for the UCB_1 algorithm are:

- an iteration index initialized at 0: `it`,
- a table of size N (the number of channels, 3 in this implementation example, but it could be arbitrarily high) for the number of times each channel has been chosen, representing T_k of (2): `Tk[]`.
- another table of size N for the empirical mean of success of each channel, i.e., $X_k(t)$ of (3): `Xk[]`.

From the point of view of the learning algorithm, a success occurs when an IoT device receives an ACK from the IoT network (as explained above), which means that the currently used frequency channel suffered no collision in both UL and DL. Otherwise, a failure occurred. The update of the selected channel empirical means X_k is reconstructed easily from the number of activations and the previously stored X_k value. Therefore, it is not necessary to store in memory the results of all past iterations, but just only a summary of it (its mean). The proposed solution is thus realistic and efficient, as it only requires a bounded storage capacity.

Then, after an initialization phase where each channel is selected alternatively once, the channel selection really starts to use the UCB_1 indexes [4]. It consists for each iteration in choosing the frequency channel with the greatest index B_k as defined in (5), that is computed for each channel like this in a *for loop* on *i* index, and with *alpha* the UCB_1 parameter α that controls the exploration vs. exploitation trade-off [4]:

```
Ak[i] = sqrt(alpha * log(it) / Tk[i])
```

The IoTligent device then selects the channel having the greatest UCB_1 index B_k [4]:

```
for i in range(N):
  Bk[i] = Xk[i] + Ak[i]
  if Bk[i] > bestChannel:
     bestChannel = Bk[i] ; freq = tabFreq[i]
```

E. Results for the second measurement

Experiments have been done on a real LoRa network currently deployed with K=3 channels. We present results obtained on an IoTligent device, for 129 transmissions done every 2 hours, for a period of 11 days. Fig. 8 shows the evolution of the T_k index through time, which is the number of time each channel has been selected by the learning algorithm. In the figures, the red curve is for channel #0 (at 868.1 MHz), the green curve is for channel #1 (868.3 MHz) and the light blue curve is for channel #2 (868.5 MHz).

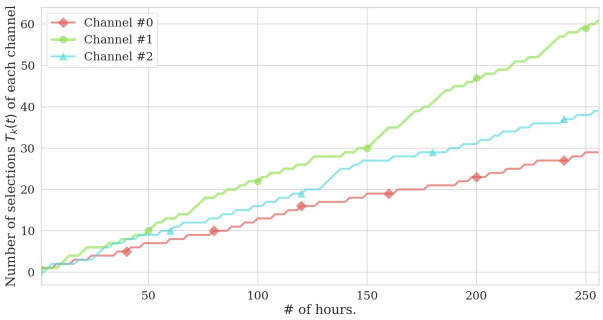

Fig. 8. Evolution of the T_k index through time, as learning happens.

Fig. 9 gives the empirical mean *Xk* experienced by the device on each of the 3 channels. Each peak corresponds to a successful LoRa bi-directional exchange between the device and the AS: from the device uplink transmission to the ACK reception (downlink) by the device.

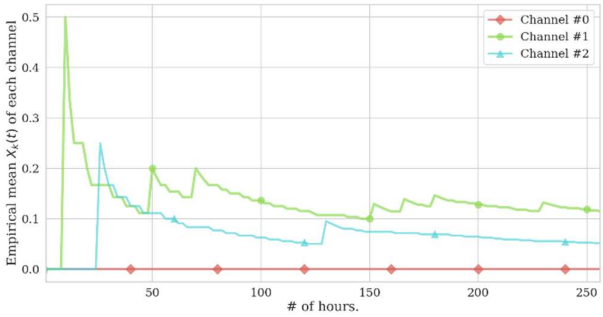

Fig. 9. Evolution of the X_k empirical mean through time.

We can see that channel #1 gives the best results, before channel #2, but channel #0 always failed in sending back an ACK to the device. Each peak in Fig. 8 reveals a successful case where an ACK has been received by IoTligent device. Fig. 9 gives the end results after 11 days. We can see that channel #0 has been tried 29 times with $S_k[0] = 0$ success (i.e., no ACK received by the device). So the learning algorithm made the device use 61 times channel #1 with $S_k[1] = 7$ successful bi-directional exchanges, and 39 times channel #2 with $S_k[2] = 2$ successes. This corresponds to 7 (respectively 2) peaks of $X_k[1]$ (respectively $X_k[2]$) on Fig. 2.

TABLE I. RESULTS AT THE END OF THE EXPERIMENT

$T_k[0] = 29$	$T_k[1] = 61$	$T_k[2] = 39$
$X_k[0] = 0.0$	$X_k[1] = 0.115$	$X_k[2] = 0.051$
$S_k[0] = 0$	$S_k[1] = 7$	$S_k[2] = 2$

The empirical mean gives the vision the device obtained from the channels, i.e., a mean probability of 11.5% of successful bi-directional connection for channel #1 and 5% for channel #2, whereas channel #0 never worked from the device point of view. With a normal device, i.e. a non IoTligent device, that uses a purely random access, trying once over 3 times on each channel, for a global average successful rate of 5.5%.

It is important to note that here the learning algorithm is mostly in its exploration phase, but it is learning very fast. Only during the last 2 days of the experiment, channel #1 has already been used 4 times more than channel 0 and 2.5 times more than channel #2, which means that learning is already very effective. As proven for UCB algorithms [5][7], channel 1 will be more and more selected so that the global success rate will converge to the percentage of success of the best channel, which is 11.5% in this experiment (this estimate can be considered as a good evaluation as it is based on 61 trials). In other words, this means that a mean of 15 successes can be expected in the long term over the same period of 11 days with IoTligent. On the contrary, normal devices will never improve and stay in the current average, i.e. in average 7 successful transmissions on the same period duration.

In order to have the same rate of successful transmissions, normal IoT devices should consequently transmit twice *more* often, which has two negatives impacts. The first impact is that normal IoT devices autonomy will be twice *less* than IoTligent devices. The second but not the least impact is that devices will occupy twice more times the radio channels, hence contributing to increase even more the risks of radio collisions and thus the IoT bands congestion.

VII. MEASUREMENT 3 : IOTLIGENT OPERATION IN A LORAWAN NETWORK WITH EMULATED ARTIFICIAL TRAFFIC

As a way to make a complete validation of our proposal, we now propose to combine the two previous experiments, by running IoTligent real LoRa IoT devices on a real LoRaWAN network, but under the future expected load, emulated using USRP platforms.

A. Experimental setup

As far as we know, this is the first evaluation in a real LoRaWAN network of LoRa devices running on-line learning algorithms, with emulated traffic reproducing very dense IoT conditions. The measures use a Faraday cage and an anechoic chamber, in order to avoid jamming real LoRaWAN networks operating in the surroundings of the laboratory. It also enables to be fully in control of the ISM jammers and channel occupancy, and to perfectly monitor what is happening during the measurement campaigns. As for the first PoC measurement, we use one (or several) USRP platform as a traffic generator, in order to emulate the traffic generated by the surrounding IoT devices. Each channel's occupancy load can be set independently on demand, so that it is non uniform over the channels. The experiments presented below used a set of K=7 channels, with different colors in the next plots:

- Channel #0 : 866.9 MHz, in red,
- Channel #1 : 867.1 MHz, in orange,
- Channel #2 : 867.3 MHz, in light green,
- Channel #3 : 867.5 MHz, in green,
- Channel #4 : 867.7 MHz, in light blue,
- Channel #5 : 867.9 MHz, in dark blue,
- Channel #6 : 868.1 MHz, in purple.

For each experiment, we compare the results of two LoRa IoT devices: (i) one *IoTligent device* running the learning algorithm (UCB_1) ; (ii) one usual LoRa device that acts as a reference and that we name *reference IoT device*. The gain of our approach can be measured by the difference between the number of successful communications obtained by IoTligent device compared to the results of the reference IoT device, as both run in the same conditions of traffic load. It can also be made by a comparison of their success rate.

During the experiments, we make devices transmit every 20 seconds. A successful communication occurs when the IoT device receives in DL an ACK to its own last UL transmission (on the same channel). In that case, the gateway received the

transmissions, on the frequency channel selected by the devices, i.e. randomly for the reference device, or by running the bandit algorithm for the IoTligent device. The gateway then forwards the message to the Application Server through the LoRa Network Server. The acknowledgment is then sent back the opposite way and the gateway uses the same channel as the one used by the device in uplink, regardless if the device is IoTligent or not. As only these two devices are requesting acknowledgements, and no other real LoRa device can access the gateway in the chamber, the constraint of 1% duty cycle is not exceeded by the gateway in any channel.

We ran the experiments over several hundreds of iterations (i.e., of transmissions) so that they have a duration of a couple of hours. We used USRP platforms as jammers that generate emulated IoT traffic. As the USRP transmission power, for jamming signal, is not as high as those of LoRa IoT devices and LoRa gateway, LoRa IoT devices power has been decreased with attenuators of 40 dB. However this has not been possible to do on the gateway, and we discuss the consequence below.

For one device, when no acknowledgement is received, it means that there has been a collision either in DL or in UL. Here we can assert that collisions only occur in UL as we can check that at each time a message has been received by the AS, an ACK has been correctly received by the IoT device (reference or IoTligent). This is because the gateway DL feedback power is very high compared to the USRP jamming level.

We now detail here a couple of scenarios that have been executed for measurement 3, one with a medium density context of IoT devices, and second one with even more dense conditions.

B. *Scenario 1: Not too heavy traffic and one free channel*

We choose in scenario 1 a context where channels occupancy is slightly decreasing from one to another. This enables to understand how the algorithm runs as a first approach. The percentage of occupancy for each channel is 30% for channel #0, 25% for channel #1, and so on until 0% for channel #6, so a vector like this for the K=7 channels: {0.30, 0.25, 0.20, 0.15, 0.10, 0.05, 0}. So the channel where less radio collisions should occur is obviously channel #6 and we name it the *best channel*.

We visualize in Fig. 10 below the number of times each of the channels has been used by IoTligent device through time. The dark green curve of channel #6 is clearly leading the race, followed far away by light blue channel #5, and then purple channel #4, and so on. It allows to conclude that the learning algorithm has clearly been able to favor the use of the less occupied channels by the LoRa IoTligent device. Moreover, after a short time, it has been able to make a difference between each different level of occupancy in the channels, with a great advantage to the best one.

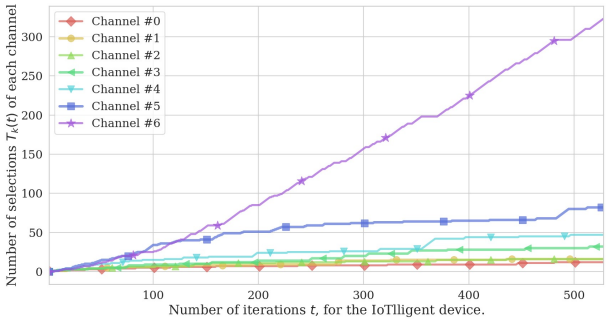

Fig. 10. Evolution of the number of selections through time of IoTligent device for scenario 1.

To give a rigorous comparison, we can see on Fig. 11 that the reference LoRa device, on its side, has uniformly used the 7 channels during the experiment. This perfectly illustrates the difference between an IoTligent and a usual (naïve) IoT device. IoTligent device uses Reinforcement Learning to make the choice of a channel before each transmission, based on a given metric (reward) depending on its past experience. Here the followed policy is UCB_1 (with exploration factor $\alpha = 2$).

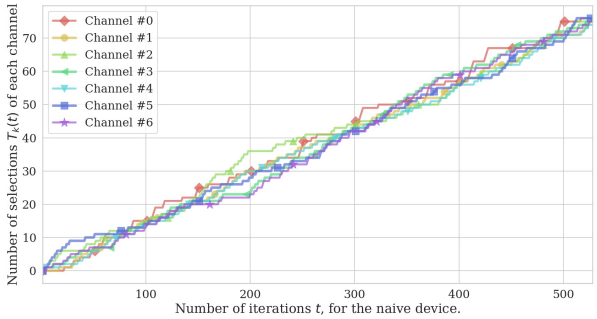

Fig. 11. Evolution of the number of selection through time, for the reference (naive) IoT device for scenario 1.

The curves of Fig. 12 represent the empirical mean reward of (3) for each channel obtained by IoTligent device. With no surprise, the channels that have been the most used for transmissions are those with the best mean reward, i.e. the best percentage of vacancy. The same order is found as in Fig. 10, with dark green curve of channel #6 first, followed by light blue channel #5, and then purple channel #4, and so on. Indeed, Fig. 10 is a consequence of Fig. 12, as UCB_1 chooses more and more with time, as stated in (5), the channels with higher empirical means X_k, as the A_k term decreases with the number of activation T_k and becomes negligible compared to X_k. Note that jumps in the curves are having bigger steps on the channel curves that have been used less often, as A_k of (5) is still predominant when T_k is low.

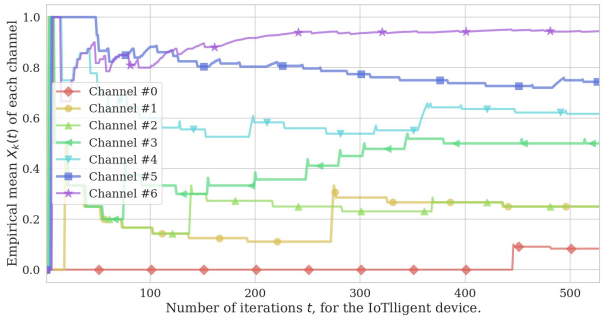

Fig. 12. Evolution of the X_k empirical mean through time of IoTligent device for scenario 1.

We also monitor the reference device empirical mean reward in Fig. 13, but just for comparison purposes as it is not used by the reference IoT device to select its channels. The difference with the IoTligent device is that reference device has played more times the bad channels and less times the best channels. As a consequence, there is a little bit more variance on the empirical mean reward for less occupied channels for the reference IoT than for the IoTligent device, but the opposite for most occupied channels. However the goal is not to recover the empirical mean from the cognitive radio point of view, but to order the channels in terms of their occupancy rate.

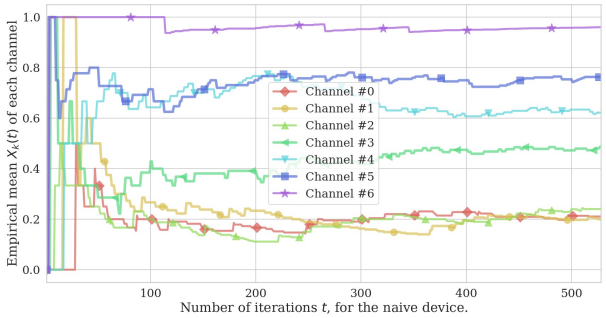

Fig. 13. Evolution of the X_k empirical mean through time of reference IoT device for scenario 1.

In TABLE IV. are listed the number of activations and percentages of successful transmission obtained on each channel by the IoTligent and reference IoT devices. Whereas the reference IoT device uniformly transmits on all channels, we can see that the IoTligent device has been concentrating on most vacant channels, with a clear choice for the less occupied channel #6. Over 526 iterations, 323 transmission have been done in this channel, i.e., more than 4 times compared to the reference IoT, and almost 27 times more than for channel #0 for IoTligent device. This gives the opportunity to the IoTligent device to increase drastically its global successful rate that can be seen on TABLE III. The IoTligent solution allows the device to reach a successful transmission rate of almost 80 % against 50 % for the reference IoT device.

TABLE II. COMPARED RESULTS AT THE END OF THE EXPERIMENT BETWEEN REFERENCE IOT DEVICE AND IOTLIGENT DEVICE IN TERMS OF NUMBER OF ACTIVATIONS AND PERCENTAGE OF SUCCESS ON EACH CHANNEL FOR SCENARIO 2.

	Reference IoT		**IoTligent**	
Channel	**% of success**	**Nb of activations**	**% of success**	**Nb of activations**
#0	21 %	76	8 %	12
#1	20 %	76	25 %	16
#2	24 %	75	25 %	16
#3	49 %	76	50 %	32
#4	62 %	74	61,7 %	47
#5	76,3 %	76	74,4 %	82
#6	96 %	75	94,4 %	**323**

TABLE III. emphasizes the advantage of IoTligent solution as it almost improves by 2.5 times the performance of reference device, in terms of use of the best channel. This is due to the ability to favor the use of less occupied channels, and especially channel #6 which is completely vacant.

TABLE III. COMPARED RESULTS AT THE END OF THE EXPERIMENT BETWEEN REFERENCE IOT DEVICE AND IOTLIGENT DEVICE IN TERMS OF PERCENTAGE OF SUCCESS (I.E. ACK RECEIVED BY THE DEVICE) FOR SCENARIO 2.

	Reference IoT	**IoTligent**
Nb of iterations	528	528
Nb of no ACK	266	108
% of success	49,6 %	79,5 %

C. *Scenario 2: Very heavy traffic*

Let us consider now a heavy traffic scenario as percentage of occupancy for each channel is set to 40% for channel #0, #1 and #2, 30% for channel #3, 20% for channel #4, 15% for channel #5 and 10% for channel #6. So once again, channel #6 is the *best channel*.

As in the previous scenario, we illustrate on Fig. 14 that channel #6 is the most played one, even if it is not fully vacant. Indeed, the learning algorithm takes into account the relative occupancy rate between the channels. So the differences with the three following channels (#5, #4, #3) look like scenario 1, but clearly all the three first channels are almost always avoided.

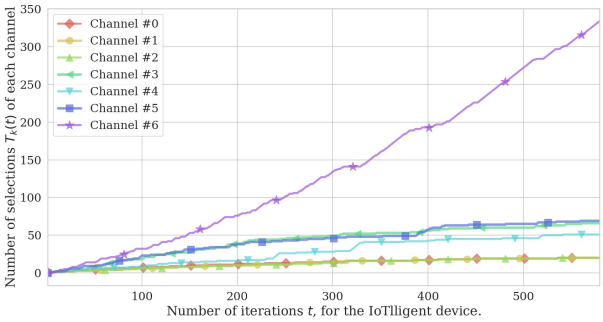

Fig. 14. Evolution of the *number of selection* through time of IoTligent device for scenario 2.

We see the empirical mean measured by the IoTligent device, in Fig. 15. It reflects once again the (inverse) order of the occupancy rate that has been set for the channels.

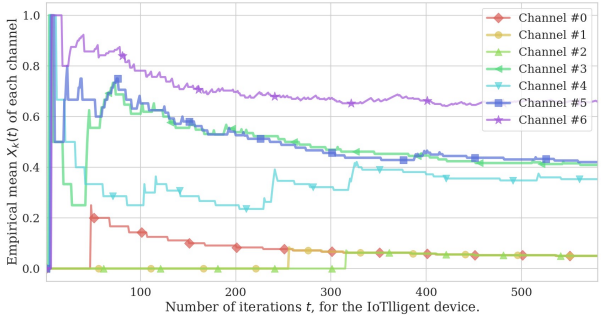

Fig. 15. Evolution of the X_k empirical mean through time of IoTligent device for scenario 2.

Fig. 16 confirms the results in terms of mean success of channels by the reference IoT device, and we find the same channel mean occupancy rates. Results are even more solid for the reference device as all channels have been selected the same number of times. Here also, the IoTligent device has only gathered a few samples on the worst channels so that the evaluated empirical mean is not so realistic.

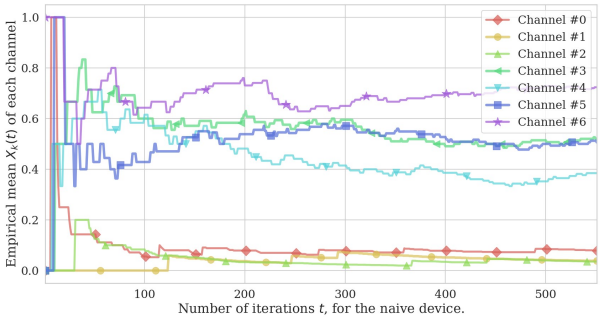

Fig. 16. Evolution of the X_k empirical mean through time of reference IoT device for scenario 2.

TABLE IV. shows the number of times each channel has been selected and the obtained empirical percentage of successful transmission. For both devices, we can see a direct (inverse) correspondence with the occupancy rate of each channel given earlier. Most importantly, we observe an unbalanced number of activations of the channels thanks to the learning algorithm. Whereas the reference device roughly uses each channel equally (around 80 times), we can see that the IoTligent device concentrates its transmission in channel #6 (334 times) as it provides the best percentage of success, and neglects the worse channels (20 times).

TABLE IV. COMPARED RESULTS AT THE END OF THE EXPERIMENT BETWEEN REFERENCE IOT DEVICE AND IOTLIGENT DEVICE IN TERMS OF NUMBER OF ACTIVATIONS AND PERCENTAGE OF SUCCESS ON EACH CHANNEL FOR SCENARIO 2.

	Reference IoT		IoTligent	
Channel	**% of success**	**Nb of activations**	**% of success**	**Nb of activations**
#0	7,9 %	76	5 %	20
#1	3,9 %	78	5 %	20
#2	3,5 %	85	5 %	20
#3	52 %	77	41 %	66
#4	38,5 %	78	35 %	51
#5	50,6 %	81	42 %	69
#6	72,4 %	76	**65,9 %**	**334**

We can also see on TABLE IV. that despite the IoTligent device experimented worse results than the reference IoT device on the best channel (channel #6), its global results are much better.

Results of TABLE V. in terms of percentage of success show how efficient is the proposed solution. With a global mean of 28% of channel occupancy on the 7 channels, the reference IoT device obtains in this experiment a 32 % of transmission successes whereas IoTligent device is able to reach 51%.

TABLE V. COMPARED RESULTS AT THE END OF THE EXPERIMENT BETWEEN REFERENCE IOT DEVICE AND IOTLIGENT DEVICE IN TERMS OF PERCENTAGE OF SUCCESS (I.E. ACK RECEIVED BY THE DEVICE) FOR SCENARIO 2.

	Reference IoT	**IoTligent**
Nb of iterations	551	580
Nb of no ACK	373	283
% of success	32,3 %	51,2 %

Remark that these figures take into account the very beginning of the learning phase where the bandit algorithm is still exploring. TABLE VI. gives the results during the 100 last iterations. Percentage of success reaches 66.7 % for IoTligent indeed, which is coherent with the results of TABLE V. where we can see that channel #6 percentage of success is almost 66 %. As IoTligent is now almost only targeting channel #6, its percentage of success is slowly sliding towards the empirical mean of this channel. This is exactly what is proven to be achieved at infinity by the mathematical proof of converge of UCB_1 [4]. However we demonstrated that efficiency is obtained much earlier, in practical radio conditions.

TABLE VI. COMPARED RESULTS THE LAST 100 ITERATIONS OF THE EXPERIMENT BETWEEN REFERENCE IOT DEVICE AND IOTLIGENT DEVICE IN TERMS OF NUMBER OF ACTIVATIONS AND PERCENTAGE OF SUCCESS ON EACH CHANNEL FOR SCENARIO 2.

	Reference IoT		IoTligent	
Channel	**% of success**	**Nb of activations**	**% of success**	**Nb of activations**
#6	66,7 %	81	80 %	15

This final measurement campaign definitely confirms that the proposed approach can be a solution for radio collision mitigation in the future IoT ultra-dense networks in the unlicensed-bands. Our study in [8] confirmed by simulation that advantage still remains even if the number of IoTligent devices increases, using solution from the literature in order to orthogonalize IoTligent devices without coordination.

VIII. CONCLUSION

We describe in this paper the solution we propose to mitigate radio collisions in IoT unlicensed bands. Our solution is to use machine learning algorithms, to be implemented on the IoT device side, at a very low cost of implementation and no protocol overhead. We propose to use Multi-Armed Bandit algorithms (MAB) and we first prove the efficiency of the method on a proof-of-concept demonstration based on USRP platforms in laboratory conditions (named MALIN). We then present the implementation of these MAB learning algorithms on devices deployed in a real IoT network, and finally we show the validity in the expected future conditions of massive IoT deployment. Implementation on LoRa devices in a real LoRaWAN network is demonstrated and is named IoTligent. As far as we know, we propose the first implementation of a decentralized spectrum learning scheme for IoT wireless networks. Even if the current IoT networks are not (yet) densely populated by devices, medium and even short term forecasts predict a high number of devices to overcrowd the ISM unlicensed bands. The IoTligent approach is then a solution to extend on the one hand the IoT devices battery life,

which is a key performance indicator in any IoT ecosystem, and on the other hand to mitigate the collision issue that will occur with the growing number of IoT devices.

ACKNOWLEDGMENT

This publication is supported by the European Union through the European Regional Development Fund (ERDF), and by the French Region of Brittany, Ministry of Higher Education and Research, Rennes Métropole and Conseil Départemental 35, through the CPER Project SOPHIE / STIC & Ondes. The authors would like also to thank Rémi Bonnefoi for the MALIN implementation, as well Yalla Diop for their technical support on LoRa network and Pycom programming.

REFERENCES

[1] N. Sornin, M. Luis, T. Eirich and A. L. Beylot "LoRaWAN specification", technical report, LoRa Alliance, Inc., January 2015.

[2] C. Fourtet, "The technical challenge of future IoT networks and their consequences on modem's and SoC's design", Réseaux et services conference, R&S, Paris, France, June 2015.

[3] W. Jouini, D. Ernst, C. Moy, J. Palicot, "Multi-Armed Bandit Based Policies for Cognitive Radio's Decision Making Issues", Signal Circuits and Systems Conference, Jerba, Tunisia, 6-8, Nov. 2009.

[4] W. Jouini, D. Ernst, C. Moy and J. Palicot, "Upper Confidence Bound Based Decision Making Strategies and Dynamic Spectrum Access," *IEEE ICC, International Conference on Communications,* Cape Town, South Africa, May, 2010.

[5] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," Advances in Applied Mathematics, vol. 6, no. 1, pp. 4–22, 1985.

[6] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem", Machine Learning, volume 47, number 2-3, May 2002.

[7] C. Moy, "Reinforcement Learning Real Experiments for Opportunistic Spectrum Access", *Karlsruhe Workshop on Software Radio*, Karlsruhe, Germany, March 2014.

[8] R. Bonnefoi, L. Besson, C. Moy, E. Kaufman and J. Palicot, "Multi-Armed Bandit Learning in IoT Networks: Learning helps even in non-stationary settings", CrownCom 2017, Lisbon, September 2017.

[9] A. Anandkumar, N. Michael, A. K. Tang, and A. Swami, "Distributed algorithms for learning and cognitive medium access with logarithmic regret", IEEE Journal on Selected Areas in Communications, vol. 29, no. 4, Apr. 2011.

[10] H. Robbins, "Some aspects of the sequential design of experiments," Bulletin of the American Mathematical Society, vol. 58, no. 5, pp. 527–535, 1952.

[11] Q. Zhao, B. Sadler, "A Survey of Dynamic Spectrum Access", in IEEE Signal Processing and Magazine, May 2007.

[12] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," Biometrika, vol. 25, 1933.

[13] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of Stochastic and Non-Stochastic Multi-Armed Bandit Problems," Foundations and Trends® in Machine Learning, vol. 5, no. 1, pp. 1–122, 2012.

[14] S. Agrawal and N. Goyal, "Analysis of Thompson sampling for the Multi-Armed Bandit problem", in JMLR, Conference On Learning Theory, 2012.

[15] E. Kaufmann, N. Korda, and R. Munos, "Thompson Sampling: an Asymptotically Optimal Finite-Time Analysis", pp. 199–213. Springer, Berlin Heidelberg, 2012.

[16] V. Toldov, L. Clavier, V. Loscrí and N. Mitton, "A Thompson Sampling approach to channel exploration-exploitation problem in multihop cognitive radio networks", in PIMRC, 2016.

[17] A. Maskooki, V. Toldov, L. Clavier, V. Loscrí, and N. Mitton, "Competition: Channel Exploration/Exploitation Based on a Thompson Sampling Approach in a Radio Cognitive Environment", EWSN, 2016.

[18] C. Moy, J. Palicot, and S. J. Darak, "Proof-of-Concept System for Opportunistic Spectrum Access in Multi-user Decentralized Networks", EAI Endorsed Transactions on Cognitive Communications, volume 2, 2016.

[19] L. Besson, "SMPyBandits: an Open-Source Research Framework for Single and Multi-Players Multi-Arms Bandits (MAB) Algorithms in Python". Code on https://GitHub.com/SMPyBandits/SMPyBandits and documentation on https://SMPyBandits.GitHub.io/

[20] L. Besson, R. Bonnefoi, C. Moy, *"MALIN: Multi-Armed bandit Learning for Iot Networks with GRC*: A TestBed Implementation and Demonstration that Learning Helps", ICT 2018, France, June 2018.

[21] LoRaWAN™ v1.1 Specification, 2017, LoRa Alliance Inc, https://LoRa-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf

[22] C. Moy, "IoTligent: First World-Wide Implementation of Decentralized Spectrum Learning for IoT Wireless Networks", URSI AP-RASC, New Delhi, India, 9-14 March 2019.

[23] C. Moy and L. Besson, "Decentralized Spectrum Learning for IoT Wireless Networks Collision Mitigation", First International Workshop on Intelligent Systems for the Internet of Things, Santorini Island, Greece, May 29-31 2019.