URSI AP-RASC 2019, New Delhi, India, 09 - 15 March 2019

IoTligent: First World-Wide Implementation of Decentralized Spectrum Learning for IoT Wireless Networks

Christophe MOY
Univ Rennes, CNRS, IETR - UMR 6164, F-35000, Rennes, France, christophe.moy@univ-rennes1.fr

Abstract

We propose the first implementation of learning algorithms on LoRa devices operating in a real LoRaWAN network. The goal of learning intends here to diminish collisions with other RF signals present in the ISM band. We explain that the proposed solution, named IoTligent, add neither network overhead so that no change is required to LoRaWAN, nor processing overhead so that it can be run in the devices. Experimental measurements done in a real LoRa network show that IoTligent device battery life can be extended by a factor 2 in our experiment.

1. Introduction

We propose the first implementation of learning algorithms on devices deployed in a real LoRa network in order to make Internet of Things (IoT) devices mitigate collisions with other Radio Frequency (RF) signals present in the ISM band. The proposed implementation runs in the 868 MHz band but could be used in any other ISM band, whatever the country. Any other IoT LPWAN (Low Power Wide Area Network) standard than LoRa could be targeted as soon as channel assignation is not imposed by a central node in the network. We name this learning decentralized approach, e.g. on device: IoTligent.

2. Issue, hypothesis and decentralization pros

2.1. Collisions vs autonomy

Collisions is the main drawback of IoT in terms of battery autonomy at first level, but also of IoT viability itself in the ISM bands. Indeed collisions may cause (many) retransmissions at the cost of a lower battery lifetime and RF contention increase, or even worse a total failure of the device.

2.2. Hypothesis

IoTligent imposes no change on normal LoRaWAN protocol [1]: no extra retransmission, no extra-power to be sent, no data to be added in frames. Only condition is that the proposed solution should work with acknowledged mode for IoT. 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 jammed than others, but it is not possible to predict it in time and space, so the need to learn on field.

2.3 A device-side solution for spectrum management

IoTligent learning algorithm is a kind of artificial intelligence (AI) algorithm that is so simple to implement that it can be applied at IoT device side with almost no extra overhead (processing, memory, energy consumption). It is indeed much more efficient to implement radio collision mitigation approaches on device side as devices may be quite far from gateways, and suffer from different radio and jamming/co-existence conditions. But they are the place where watts count at transmission, and where sensitivity should be the best at reception, as no extra-processing can be afforded.

3. IoTligent: learning for non coordinated and decentralized IoT dynamic spectrum access

3.1 Proposed solution

IoTligent is based on reinforcement learning algorithms such as those already studied [2] and experimented on real radio signals for Cognitive Radio and especially Opportunistic Spectrum Access (OSA) [3]. We assert that, as for OSA, IoT spectrum access issue can be modeled as a Multi-Armed Bandit (MAB) problem. Reinforcement learning is based on a feedback loop that gives a success measure of experience. In the IoT context we propose to use the acknowledgement (ACK) sent by the gateway to the IoT device. First AI experiments on IoT radio signals have been done in laboratory conditions with USRP on emulated IoT signals [4], using Upper Confidence Bound (UCB) algorithms that have been the first derived to solve the MAB issue [5].

3.2 Advantages of the proposed solution

The main advantages of IoTligent are:

- implementation and execution require very low processing and memory overhead so that it is possible to add it in IoT devices at no money cost, negligible complexity (processing, hardware, memory) and extra-energy consumption overhead,
- using these learning algorithms never gives worse results than a state-of-the art random solution [6],
- no coordination is required between devices (see [6] and [7] for discussion about this issue),
- 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.

3.3 Analysis of collisions

Radio collisions will be the weak point of LPWAN IoT networks operating in the license free bands such as ISM bands. They may occur with other LoRa devices or any other IoT radio signals using another IoT radio standard (SigFox, Ingenu, Weightless, etc.). Each IoT standard uses indeed its own rules for channeling, bandwidth, user repartition, etc. Moreover, other radio signals present in the ISM band which are not IoT signals may also interfere. Unlicensed band indeed, does not mean un-ruled band (duty cycle, power limit, etc.), but it is by definition more exposed to the non-respect of these few rules.

4. Implementation details

The implementation of the learning algorithm we propose in this paper is decentralized, e.g. it takes place only on LoRa device side. As stated earlier, no other aspect of the LoRaWAN network is impacted. We explain briefly below the LoRaWAN network side configuration.

4.1 System description

A LoRaWAN network, as any other IoT network, can be summarized in 4 main elements:

- LoRa devices (our devices run IoTligent),
- LoRa gateway(s) receiving all LoRa radio signals in their 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 an acknowledgment to them (mandatory here).

4.2 Device side

We use a Pycom [8] composed of an Expansion Board and a LoPy module which can support LoRa wireless connectivity. Pycom is programmed in Python language. The frequency channels used in the experiments are those authorized in the country of experimentation (e.g. France). Three channels are usually used in Europe for uplink (UL) with a duty cycle of 1%: {868.1 MHz, 868.3 MHz, 868.5 MHz}. IoTligent is completely agnostic to the number of channels to be used and can be used in any country.

4.3 Network side

We have access to the LNS provided by Acklio Company. Acklio has several gateways in the town of Rennes where trials have been made. LNS sends the received messages to an AS which is a Linux server in the cloud. AS is running a Python program that enables to display data and metadata (e.g: frequency, time of reception, etc.). This programs also contains instructions in order to send an acknowledgment to the device, using in downlink (DL) the same frequency used by the device at UL.

5. Implementation

5.1 Device side

Based on-line examples [8] we use *LORAWAN* mode with an Over-The-Air-Activation (OTAA) using *app_EUI* and *app_key* keys:

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

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

Transmit channel frequency is then chosen in a set of N channels which is set here at:

```
N = 3
```

Indeed, we use standard Europe UL channels with the following frequency table:

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

IoTligent device infinite *while loop* is started, running the algorithm presented in next section and [2] in order to choose which frequency to be selected at each iteration before executing a send operation. ACK is then expected from the network side in *non blocking* mode so that when ACK is not received, devices just updates its learning data and still goes on.

5.2 Network side – Lora Network Server

Devices should be declared to LNS with the at least following information:

- *devEUI* : ID of the device obtained by executing a « *get_id.py* » program from [8] 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=125kHz.

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

5.3 Network side – Application Server

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

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

6. Learning algorithm in Pycom device

Learning algorithm used in IoTligent are (any) bandit algorithms, such as those used at first time for Cognitive Radio dynamic spectrum access in [2] and proposed in Python library [9]. We take here the example of UCB algorithm [5]. We have chosen these algorithms for their ease of implementation. Only data necessary to be stored for UCB 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: `Tk[]`
- another table of size N for the empirical mean of success of each channel: `Xk[]`

From the learning algorithm point of view, a success occurs when a device receives an ACK from the IoT network, which means that currently used frequency channel did suffer no collision both in UL and DL. Otherwise, a failure occurred. Update of selected channel empirical mean X_k is reconstructed easily from number of activations and previous X_k stored value.

Then after initialization phase where each channel is selected alternatively once, UCB algorithm really starts [2] . It consists for each iteration in choosing the frequency channel with greatest index B_k, where $B_k = X_k + A_k$, where A_k is a bias computed for each channel like this in a *for loop* on *i* index, and with *alpha* the UCB parameter that sets the exploration vs. exploitation trade-off [2]:

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

IoTligent channel selection is then on greatest B_k [2]:

```
for i in range(0,N):
   Bk[i] = Xk[i] + Ak[i]
   if Bk[i] > max:
       max = Bk[i]
       freq = tabFreq[i]
```

7. Results

Real experiments have been done on a real LoRa network currently deployed with 3 channels. More channels are expected to be used in the future, inducing no implementation difference (only 2 extra numbers to be saved by added channel). We now look at results obtained on IoTligent device, for 129 transmissions done every 2 hours, so an 11 days period. Figure 1 shows the evolution of *Tk* index through time, e.g. the number of time each channel has been selected by the learning algorithm through time. In the figures, black curve is for channel 0 (868,1 MHz), blue curve for channel 1 (868.3 MHz) and red curve for channel 2 (868.5 MHz).

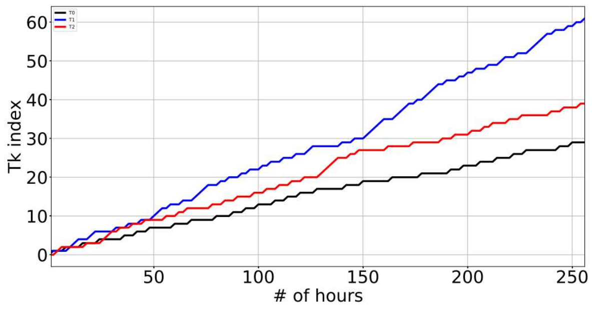

Fig. 1 – *Tk* index evolution through time

Figure 2 gives the empirical mean *Xk* experienced by the device on each of the 3 channels. Each peak corresponds to a LoRa successful bi-directional exchange between device and AS: from device transmission, to ACK reception by the device.

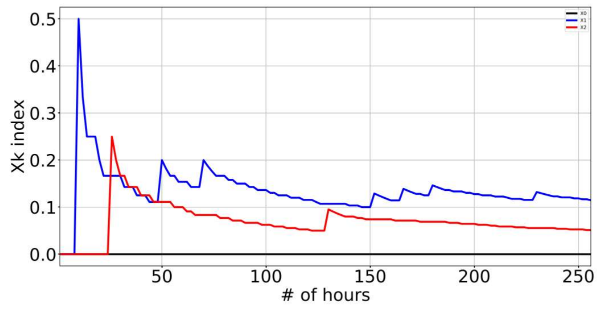

Fig. 2 – *Xk* Empirical mean evolution through time

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 figure 2 reveals a successful case where ACK has been received by IoTligent device. Figure 3 gives end results after 11 days. We can see that channel 0 has been tried 29 times with *Sk[0] = 0* success (e.g. no ACK received by the device). So learning algorithm made the device use 61 times channel 1 with *Sk[1] = 7* successful bi-directional exchanges, and 39 times channel 2 with *Sk[2] = 2* successes. This corresponds to 7 (respectively 2) peaks of *Xk[1]* (respectively *Xk[2]*) on figure 2.

Tk[0] = 29	Tk[1] = 61	Tk[2] = 39
Xk[0] = 0.0	Xk[1] = 0,115	Xk[2] = 0,051
Sk[0] = 0	Sk[1] = 7	Sk[2] = 2

Fig. 3 – Results at the end of the experiment

Empirical mean gives the vision the device obtained from the channels, e.g. 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, e.g. a non IoTligent device, a random access is done, trying once over 3 times on each channel, for a global average successful rate of 5,5%.

It is important to note that here learning algorithm is mostly in its exploration phase but is learning very fast. Only during last 2 days of the experiment indeed, 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 effective. As proven for UCB algorithms [2][3], channel 1 will be more and more selected so that global success will converge to percentage of success of the best channel which is 11,5% (this estimate can be considered as a good evaluation as it is based on 61 trials). In other words, this means that 15 success 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 current average, e.g. 7 successful transmission on the same period duration.

In order to have the same frequency of successful transmissions, normal devices should consequently transmit twice more often, which has 2 negatives impacts. First is that normal IoT devices autonomy will be twice less than IoTligent devices. Second but not least is that devices will occupy twice more radio channels, hence contributing to increase even more risks of radio collisions and IoT band congestion.

8. Conclusion

We describe in this paper the implementation of learning algorithms on devices deployed in a real IoT network. Implementation on LoRa devices in a real LoRaWAN network is demonstrated and named IoTligent. As far as we know, it is the first implementation of decentralized spectrum learning for IoT wireless networks. Even if current IoT networks are not densely populated of devices, medium and even short term forecast predict a high number of devices to overcrowd ISM unlicensed bands. IoTligent approach is then a solution to extend IoT devices battery life, which is a key performance indicator in IoT eco-system.

9. Acknowledgment

Author would like to thank Laurent Toutain from IMT Atlantique and Acklio Company, as well as Yalla Diop for their technical support on LoRa network and Pycom programming.

10. References

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

[2] 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

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

[4] 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

[5] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem", Machine Learning, vol. 47, no. 2-3, May 2002

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

[7] A. Anandkumar, N. Michael, A. K. Tang, and A. Swami, "Distributed algorithms for learning and cognitive medium access with logarithmic regret," IEEE J. Sel. Areas Commun., v. 29, no. 4, Apr. 2011.

[8] https://github.com/pycom/pycom-libraries

[9] L. Besson, "SMPyBandits: an Open-Source Research Framework for Single and Multi-Players Multi-Arms Bandits (MAB) Algorithms in Python": https://github.com/SMPyBandits/SMPyBandits