







1st MoTION Workshop - 2019: "Upper-Confidence Bound for Channel Selection in LPWA Networks with Retransmissions"

- Date  : 15th of April 2019
- Who: [Lilian Besson](#)  , PhD Student in France, co-advised by

<p><i>Christophe Moy</i> @ IETR, Rennes</p>	<p><i>Emilie Kaufmann</i> @ CNRS & Inria, Lille</p>
 <p>CentraleSupélec</p>  <p>IETR INSTITUT D'ÉLECTRONIQUE ET DE TÉLÉCOMMUNICATIONS DE RENNES</p>	 <p>Inria INVENTEURS DU MONDE NUMÉRIQUE</p>  <p>cnrs</p>

See our paper at [HAL.Inria.fr/hal-02049824](https://hal.inria.fr/hal-02049824)

Outline

1. Motivations

2. System model

3. Multi-armed bandit (MAB) model and algorithms

4. Proposed heuristics

5. Numerical simulations and results

Please 🙏 ask questions *at the end* if you want!

By R. Bonnefoi, L. Besson, J. Manco-Vasquez and C. Moy.

1. Motivations

- IoT networks are interesting and will be more and more present,
- More and more IoT objects
- \implies networks will be more and more occupied

But...

1. Motivations

- IoT networks are interesting and will be more and more present,
- More and more IoT objects
- \implies networks will be more and more occupied

But...

- Heterogeneous spectrum occupancy in most IoT networks standards
- Maybe IoT objects can improve their communication by *learning* to access the network more efficiently (e.g., by using the less occupied spectrum channel)
- Simple but efficient learning algorithm can give great improvements in terms of successful communication rates
- \implies can fit more objects in the existing IoT networks 🎉 !

2. System model

Wireless network

- In ISM band, centered at 433.5 MHz (in Europe)
- $K = 4$ (or more) orthogonal channels

One gateway, many IoT devices

- One gateway, handling different objects
- Communications with ALOHA protocol **with retransmission**
- Objects send data for 1s in one channel, wait for an *acknowledgement* for 1s in same channel, use Ack as feedback: success / failure

Transmission and retransmission model

- Each object communicates from time to time (e.g., every 10 s)
 \iff probability p of transmission at every time (Bernoulli process)
- Retransmit at most M times if first transmission failed
(until Ack is received)
- Retransmissions can use a different channel than the one used for first transmission
- Retransmissions happen after a random back-off time
back-off time $\sim \mathcal{U}(0, \dots, M - 1)$

The goal of each object

Is to *maximize* its successful communication rates

\iff *maximize* its number of received Ack.

Do we need learning for transmission? Yes!

First hypothesis

The surrounding traffic is not uniformly occupying the K channels.

Consequence

- Then it is always sub-optimal to use a (naive) uniformly random channel access
- \implies we can use online machine learning to let each IoT device learn, on its own and in an automatic and decentralized way, which channel is the best one (= less occupied) in its current environment.
- Learning is actually *needed* to achieve (close to) optimal performance.

Do we need learning for *retransmission*?

Second hypothesis

Imagine a set of IoT devices learned to transmit efficiently (in the most free channel), in one IoT network.

Question

- Then if two devices collide, do they have a higher probability of colliding again *if retransmissions happen in the same channel* ?

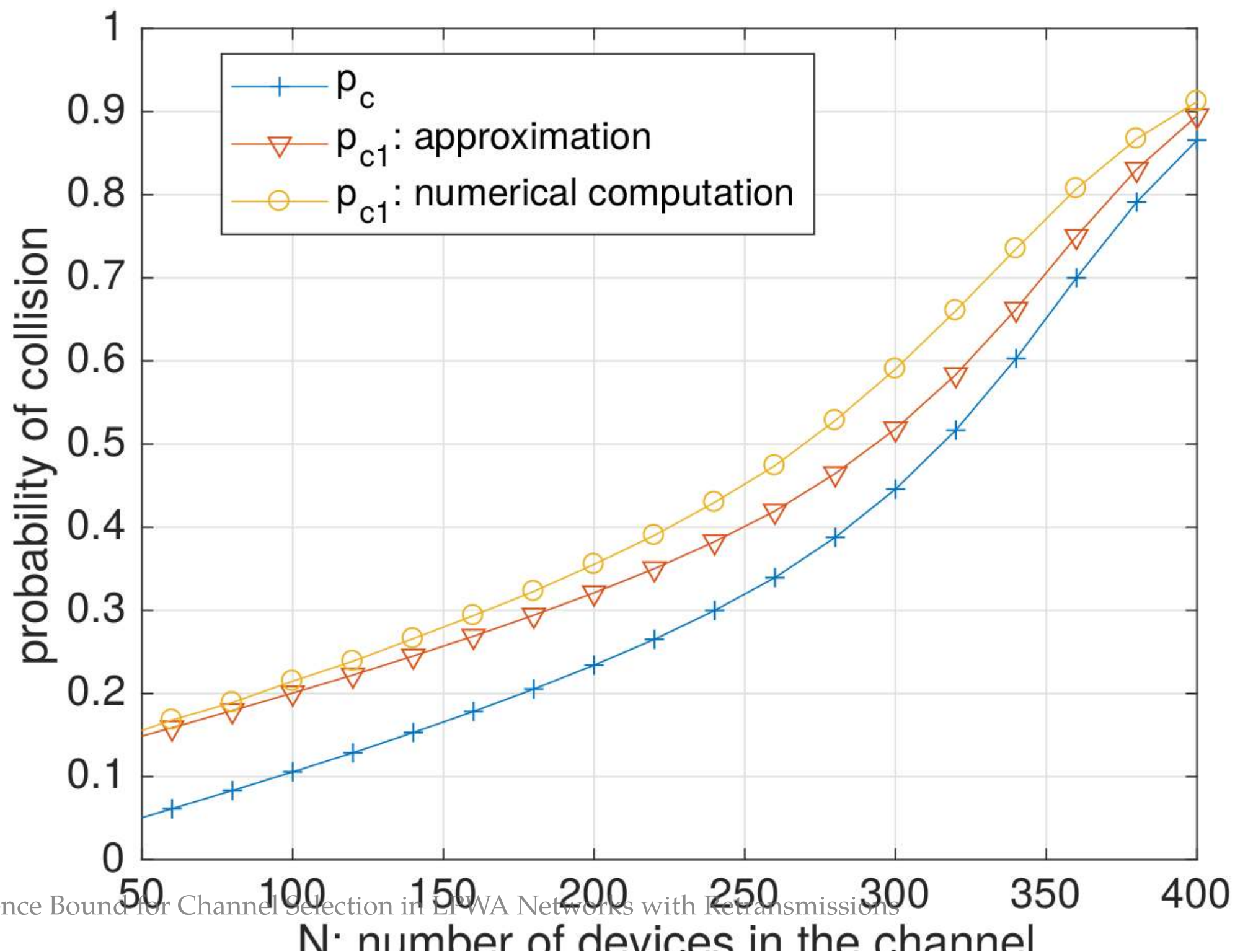
Mathematical intuition and illustration

Consider one IoT device and one channel, we consider two probabilities:

- p_c : suffering a collision at first transmission,
- p_{c1} : collision at the first retransmission (if it uses the same channel).

In an example network with...

- a small transmission probability $p = 10^{-3}$,
- from $N = 50$ to $N = 400$ IoT devices,
- \implies we ran simulations showing that p_{c1} can be more than twice of p_c (from 5% to 15%!)



Do we need learning for *retransmission*?

Yes we do!

Consequence

- Then if two devices collide, they have a higher probability of colliding again *if retransmissions happen in the same channel*
- \implies we can also use online machine learning to let each IoT device learn, on its own and in an automatic and decentralized way, which channel is the best one (= less occupied) to retransmit a packet which failed due to a collision.
- **Learning is again *needed* to achieve (close to) optimal performance.**

3. Multi-Armed Bandits (MAB)

3.1. Model

3.2. Algorithms

3.1. Multi-Armed Bandits Model

- $K \geq 2$ resources (e.g., channels), called **arms**
- Each time slot $t = 1, \dots, T$, you must choose one arm, denoted $C(t) \in \{1, \dots, K\}$
- You receive some reward $r(t) \sim \nu_k$ when playing $k = C(t)$
- **Goal:** maximize your sum reward $\sum_{t=1}^T r(t)$, or expected $\sum_{t=1}^T \mathbb{E}[r(t)]$
- Hypothesis: rewards are stochastic, of mean μ_k .
Example: Bernoulli distributions.

Why is it famous?

Simple but good model for **exploration/exploitation** dilemma.

3.2. Multi-Armed Bandits Algorithms

Often "*index based*"

- Keep *index* $U_k(t) \in \mathbb{R}$ for each arm $k = 1, \dots, K$
- Always use channel $C(t) = \arg \max U_k(t)$
- $U_k(t)$ should represent our belief of the *quality* of arm k at time t

Example: "Follow the Leader"

- $X_k(t) := \sum_{s < t} r(s) \mathbf{1}(A(s) = k)$ sum reward from arm k
- $N_k(t) := \sum_{s < t} \mathbf{1}(A(s) = k)$ number of samples of arm k
- And use $U_k(t) = \hat{\mu}_k(t) := \frac{X_k(t)}{N_k(t)}$.

Upper Confidence Bounds algorithm (UCB)

- Instead of using $U_k(t) = \frac{X_k(t)}{N_k(t)}$, add an *exploration term*

$$U_k(t) = \frac{X_k(t)}{N_k(t)} + \sqrt{\alpha \frac{\log(t)}{N_k(t)}}$$

Parameter α : tradeoff exploration vs exploitation

- Small α : focus more on **exploitation**,
- Large α : focus more on **exploration**,
- Typically $\alpha = 1$ works fine empirically and theoretically.

Upper Confidence Bounds algorithm (UCB)

for $t = 1, \dots, T$ **do**

For each channel k , $U_k(t) = \widehat{\mu}_k(t) + \sqrt{\alpha \log(t) / N_k(t)}$;

Transmit in channel $C(t) = \arg \max_{1 \leq k \leq K} U_k(t)$;

Reward $r_{C(t)}(t) = 1$, if *Ack* is received, else 0;

Update $N_k(t+1)$ and $\widehat{\mu}_k(t+1)$ for each channel k ;

end

Algorithm 1: The UCB algorithm for channel selection.

4. We Study Different Heuristics (5)

- They all use one UCB algorithm to decide the channel to use for first transmissions of any message
- They use different approaches for retransmissions:
 - "Only UCB": use same UCB for retransmissions,
 - "Random": uniformly random retransmissions,
 - "UCB": use another UCB^r for retransmissions (no matter the channel for first transmission),
 - "K-UCB": use K different UCB^j for retransmission after a first transmission on channel $j \in \{1, \dots, K\}$,
 - "Delayed UCB": use another UCB^d for retransmissions, but launched after a delay Δ .

4.0. Only UCB

Use the same UCB to decide the channel to use for any transmissions, regardless if it's a first transmission or a retransmission of a message.

for $t = 1, \dots, T$ **do**

For each channel k , $U_k(t) = \widehat{\mu}_k(t) + \sqrt{\alpha \log(t) / N_k(t)}$;

Transmit in channel $C(t) = \arg \max_{1 \leq k \leq K} U_k(t)$;

Reward $r_{C(t)}(t) = 1$, if *Ack* is received, else 0;

Update $N_k(t + 1)$ and $\widehat{\mu}_k(t + 1)$ for each channel k ;

end

Algorithm 1: The UCB algorithm for channel selection.

4.1. UCB + Random Retransmissions

```
for  $t = 1, \dots, T$  do  
  | if First packet transmission then  
  |   Use first-stage UCB.  
  | else // Random retransmission  
  |   Transmit in channel  $C(t) \sim \mathcal{U}(1, \dots, K)$ ;  
  | end  
end
```

Algorithm 2: Uniform random retransmission.

4.2. UCB + a single UCB for Retransmissions

```
for  $t = 1, \dots, T$  do
  if First packet transmission then
    | Use first-stage UCB.
  else // Packet retransmission with UCBr
    | Compute  $U_k^r(t) = \widehat{\mu}_k^r(t) + \sqrt{\alpha \log(t) / N_k^r(t)}$ ;
    | Transmit in channel  $C^r(t) = \arg \max_{1 \leq k \leq K} U_k^r(t)$ ;
    | Reward  $r_{C^r(t)}^r(t) = 1$ , if Ack is received, else 0;
    | Update  $N_k^r(t+1)$  and  $\widehat{\mu}_k^r(t+1)$  for each channel  $k$ ;
  end
end
```

Algorithm 3: UCB for retransmission.

4.3. UCB + K UCB for Retransmissions

```
for  $t = 1, \dots, T$  do           // At every time step
  if First packet transmission then
    | Use first-stage UCB.
  else // Packet retransmission with UCBj
    |  $j \leftarrow$  last channel selected by first-stage UCB;
    | Compute  $U_k^j(t) = \widehat{\mu}_k^j(t) + \sqrt{\alpha \log(t)/N_k^j(t)}$ ;
    | Transmit in channel  $C^j(t) = \arg \max_{1 \leq k \leq K} U_k^j(t)$ ;
    | Reward  $r_{C^j(t)}^j(t) = 1$  if Ack is received, else 0;
    | Update  $N_k^j(t+1)$  and  $\widehat{\mu}_k^j(t+1)$  for each channel  $k$ ;
  end
end
```

Algorithm 4: K different UCBs for retransmission.

4.4. UCB + Random Retransmission

```
for  $t = 1, \dots, T$  do // At every time step
  if First packet transmission then
    | Use first-stage UCB.
  else if  $t \leq \Delta$  then // Random selection
    | Transmit in channel  $C(t) \sim \mathcal{U}(1, \dots, K)$ .
  else // Delayed UCBd
    | Compute  $U_k^d(t) = \widehat{\mu}_k^d(t) + \sqrt{\alpha \log(t) / N_k^d(t)}$ ;
    | Transmit in channel  $C^d(t) = \arg \max_{1 \leq k \leq K} U_k^d(t)$ ;
    | Reward  $r_{C^d(t)}^d(t) = 1$  if Ack is received, else 0;
    | Update  $N_k^d(t+1)$  and  $\widehat{\mu}_k^d(t+1)$  for each channel  $k$ ;
  end
end
```

Algorithm 5: Delayed UCB for retransmission.

5. Numerical simulations and results

What

- We simulate a network,
- With many IoT dynamic devices.

Why

- They implement the UCB learning algorithm to learn to optimize their *first* transmission of any uplink packets,
- And the different heuristic to (try to) learn to optimize their *retransmissions* of the packets after any collision.

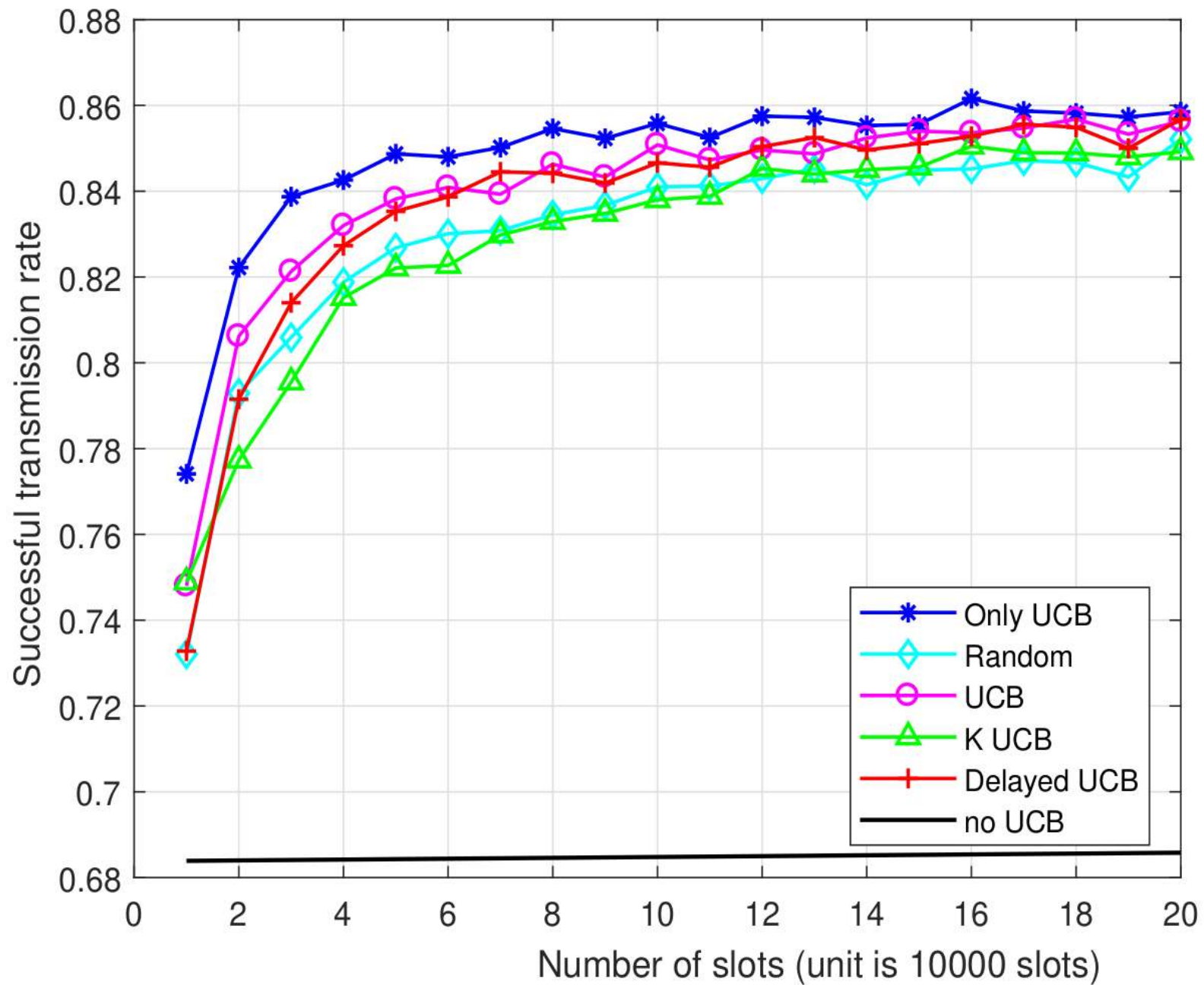
5.1. First experiment

We consider an example network with...

- $K = 4$ channels (e.g., like in LoRa),
- $M = 5$ maximum number of retransmission,
- $m = 5$ maximum back-off interval,
- $p = 10^{-3}$ transmission probability,
- $5 = 20 \times 10^4$ time slots,
- from $N = 1000$ IoT devices.

Non uniform occupancy of the 4 channels:

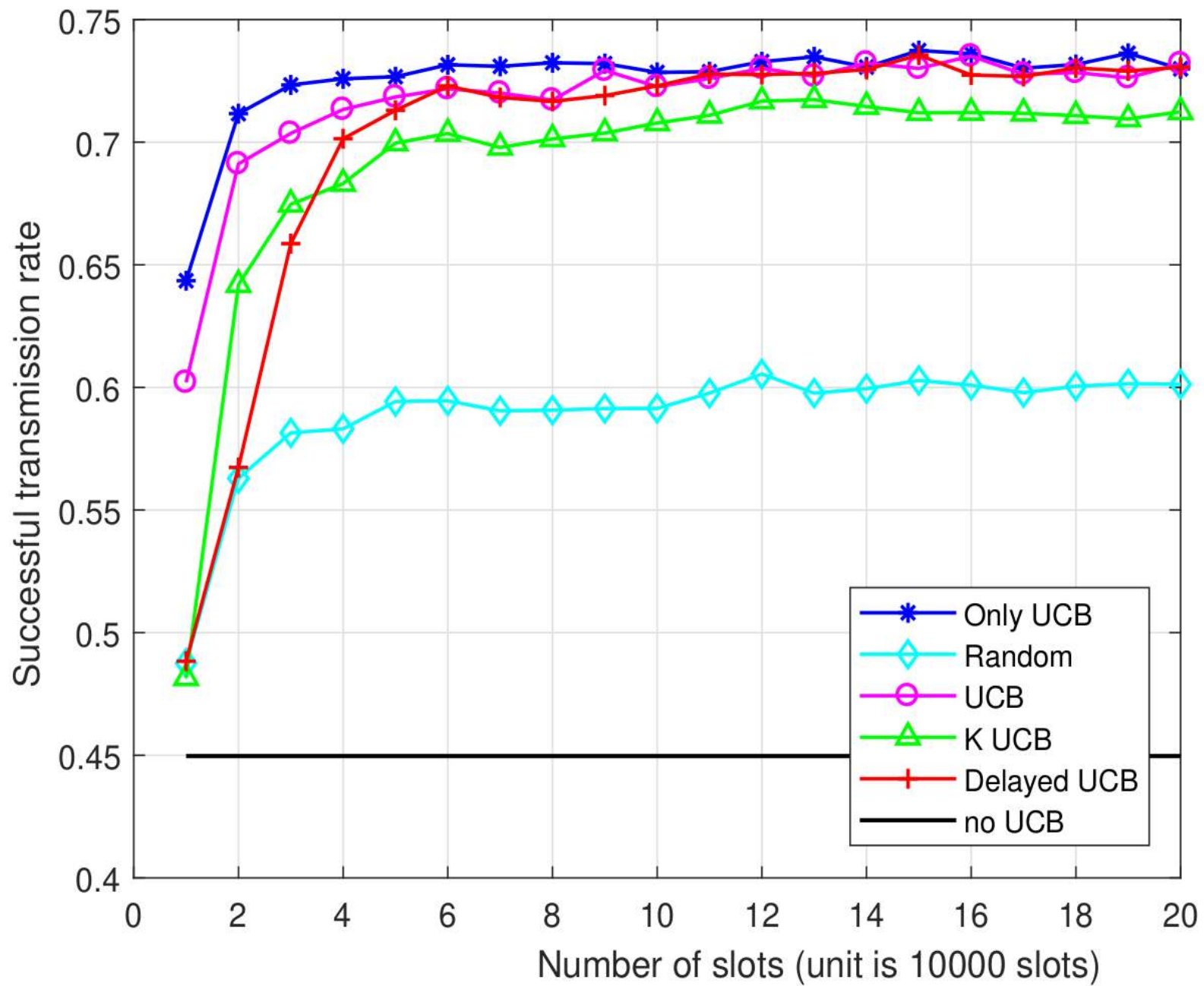
they are occupied 10, 30, 30 and 30% of times (by other IoT networks).



5.2. Second experiment

Non uniform occupancy of the 4 channels:

they are occupied 40, 30, 20 and 30% of times (by other IoT networks).



6. Summary (1/3)

Settings

1. For **LPWA networks** based on an **ALOHA protocol** (slotted both in time and frequency),
2. We presented a **retransmission model**,
3. Dynamic **IoT devices** can use **simple machine learning algorithms**, to improve their successful communication rate when accessing the network,
4. We focus on the packet retransmissions upon radio collision, by using simple and low-cost **Multi-Armed Bandit** algorithms, like **UCB**.

6. Summary (2/3)

We presented

Several learning heuristics

- that try to learn how to transmit and retransmit in a smarter way,
- by using the classical UCB algorithm for **channel selection for first transmission**: it has a **low memory and computation cost**, easy to add on an embedded CPU of an IoT device,
- and different ideas based on UCB for the retransmissions upon collisions, that add no cost/memory overhead.

6. Summary (3/3)

We showed

- Using machine learning for the *transmission* is **needed** to achieve optimal performance, and can lead to significant gain in terms of successful transmission rates (up-to 30% in the example network).
- Using machine learning for the *retransmission* is also useful, and improves over previous approach unaware of retransmission.
- The proposed heuristics outperform a naive random access scheme.
- Surprisingly, the main take-away message is that a simple UCB learning approach, that retransmit in the same channel, turns out to perform as well as more complicated heuristics.

6. Future works

- Implement our proposed approach in a real-world demo
For instance using USRP boards.
- Study a real IoT LPWAN protocol (e.g., LoRa)
- Explore in LoRa how to use machine learning (e.g., Multi-Armed Bandit algorithms) to let IoT devices learn on their own the best retransmission pattern to follow in a given scenario.

More ?

↪ See our paper: [HAL.Inria.fr/hal-02049824](https://hal.inria.fr/hal-02049824)

 Please ask questions !

Thanks for listening !