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

- *Date* 📅 : 15th of April 2019
- *By* 👋 : Lilian Besson, PhD Student in France, co-advised by

CentraleSupélec
IETR
INSTITUT D'ÉLECTRONIQUE ET DE TÉLÉCOMMUNICATIONS DE RENNES
Inria
INVENTEURS DU MONDE NUMÉRIQUE

Christophe Moy	*Emilie Kaufmann*
@ Univ Rennes 1 & IETR, Rennes	**@ CNRS & Inria, Lille**

> See our paper at `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 (the Internet of Things) is the most promizing new paradigm and business opportunity of modern wireless telecommunications,
- 📈 More and more IoT devices are using unlicensed bands
- $\Longrightarrow$ networks will be more and more occupied 💥

But…

1. Motivations

- $\Longrightarrow$ networks will be more and more occupied 💥

But...

- Heterogeneous spectrum occupancy in most IoT networks standards
- Simple but efficient learning algorithm can give great improvements in terms of successful communication rates
- IoT can improve their battery lifetime and mitigate spectrum overload thanks to learning!
- $\Longrightarrow$ can fit more devices in the existing IoT networks 🎉 !

2. System model

Wireless network

- In unlicensed bands, like the ISM bands
- $K = 4$ (or more) orthogonal channels

One gateway, many IoT devices

- One gateway, handling different devices
- Using a slotted ALOHA protocol **with retransmissions**
- Devices send data in one channel ($\nearrow$ uplink), wait for an *acknowledgement* ($\swarrow$ downlink) in same channel, use Ack as feedback : success / failure

Transmission and retransmission model

- Each device communicates from time to time (e.g., every hour)
  $\iff$ probability p of transmission at every time (Bernoulli process)

- Retransmit at most M times if first transmission failed
  (until Ack is received). (Ex. $M = 10$)

- Retransmissions can use a different channel that
  the one used for first transmission

- Retransmissions happen after a random back-off time
  back-off time $\sim \mathcal{U}(0, \cdots, m-1)$ (Ex. $m = 10$)

The goal of each device

Is to *max*imize its successful communication rates
$\iff$ *max*imize 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
- $\Longrightarrow$ 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 *re*transmission?

Second hypothesis

Imagine a set of IoT devices learned to transmit efficiently
(in the most free channels), 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

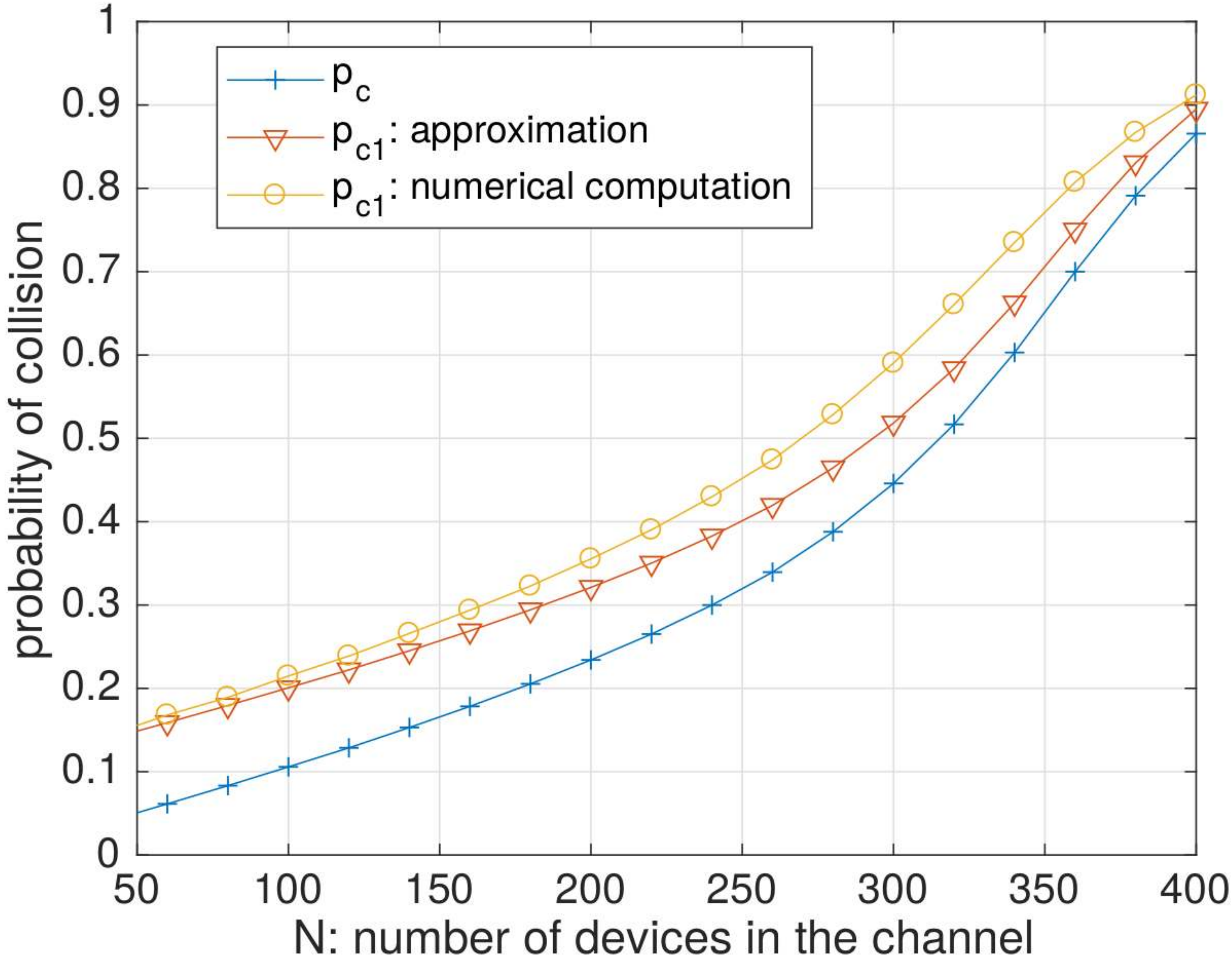
Consider one IoT device and one channel, we consider two probabilities:

- 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,

- $\Longrightarrow$ we ran simulations showing that
  p_{c1} can be more than twice of p_c (from 5% to 15%!)



p_c
p_{c1}: approximation
p_{c1}: numerical computation
probability of collision
N: number of devices in the channel
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
50
100
150
200
250
300
350
400

Do we need learning for *re*transmission?

👉 **Maybe we do!**

Consequence

- Then if two devices collide, they have a higher probability of colliding again *if retransmissions happen in the same channel*
- $\Longrightarrow$ 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 *maybe 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 $t = 1, \ldots, T$, you must choose one arm, denoted $C(t) \in \{1, \ldots, K\}$
- You receive some reward $r(t) \sim \nu_k$ when playing $k = C(t)$
- **Goal:** maximize your sum reward $\sum\limits_{t=1}^{T} r(t)$
- 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

(💥 unefficient) Example: "Follow the Leader"

- $X_k(t) := \sum_{s<t} r(s) \mathbf{1}(C(s) = k)$ sum reward from arm k
- $N_k(t) := \sum_{s<t} \mathbf{1}(C(s) = k)$ 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 $U_k(t) = \hat{\mu}_k(t) = \frac{X_k(t)}{N_k(t)}$, 👌 add an *exploration term*

$$U_k(t) = \mathrm{UCB}_k(t) = \hat{\mu}_k(t) + \sqrt{\alpha \frac{\log(t)}{N_k(t)}}$$

Parameter α = trade-off exploration *vs* exploitation

- Small $\alpha \iff$ focus more on **exploitation**,
- Large $\alpha \iff$ 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 $j \in \{1, \cdots, K\}$,
  - "Delayed UCB": use another UCB^d for retransmissions, but launched after a delay Δ.

4.1. 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.

4.2. UCB + random retransmissions

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

Algorithm 2: Uniform random retransmission.

4.3. UCB + one UCB^r for retransmissions

for $t = 1, \dots, T$ **do**
  if *First packet transmission* **then**
    Use first-stage UCB.
  else // Packet retransmission with UCB^r
    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 $r^r_{C^r(t)}(t) = 1$, 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.

4.4. UCB + $K \neq \mathrm{UCB}^j$ 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` UCB^j
        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 $r^j_{C^j(t)}(t) = 1$ 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.

4.5. UCB + Delayed UCB^d for retransmissions

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 UCB`^{d}
    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 $K = 4$ orthogonal channels,
- With many IoT dynamic devices.

Why ?

- IoT devices 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.

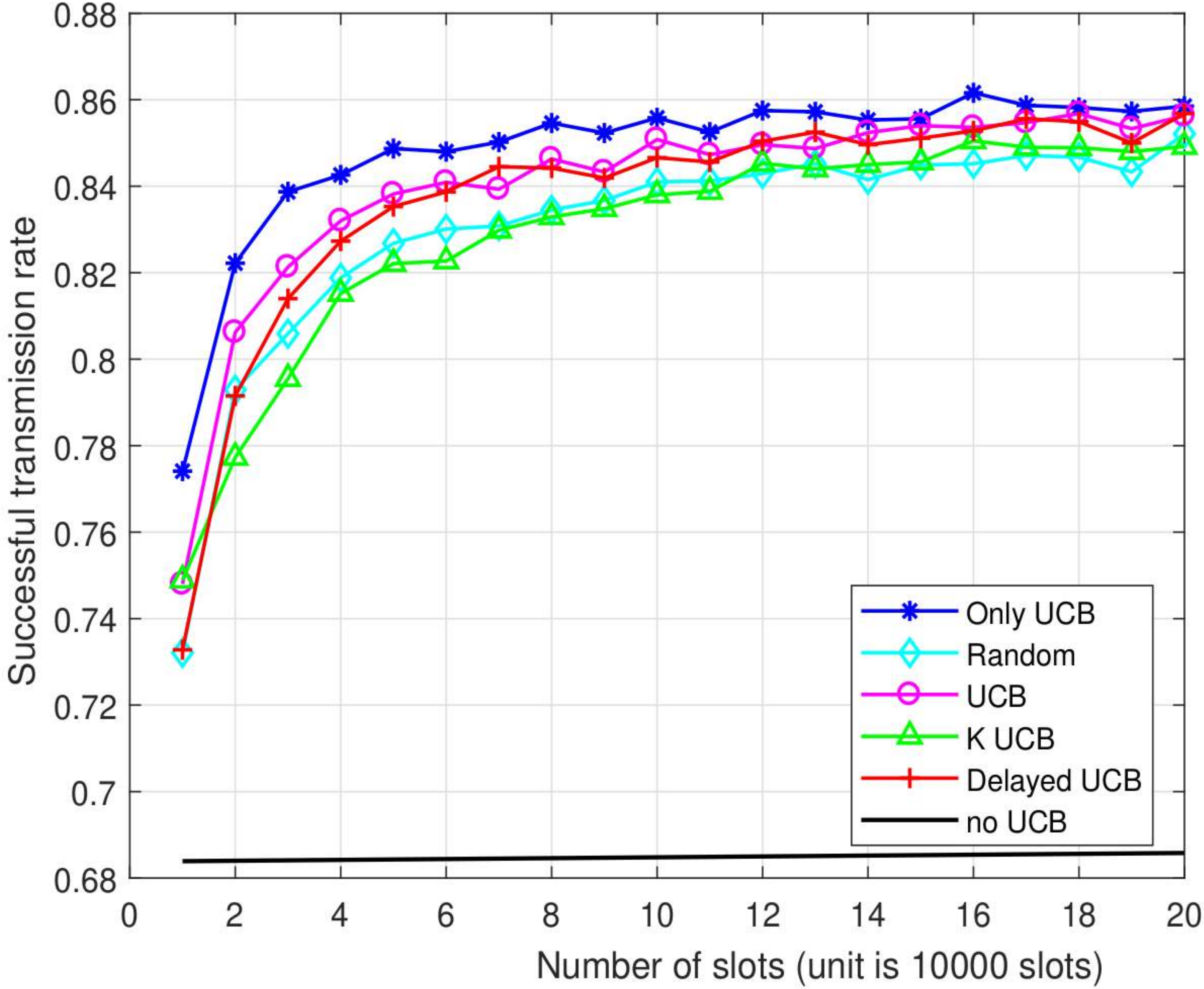
5.1. First experiment

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,
- for$N = 1000$ IoT devices.

Hypothesis

👉 Non uniform occupancy of the 4 channels:
they are occupied $10, 30, 30$ and 30% of times (by other IoT networks).

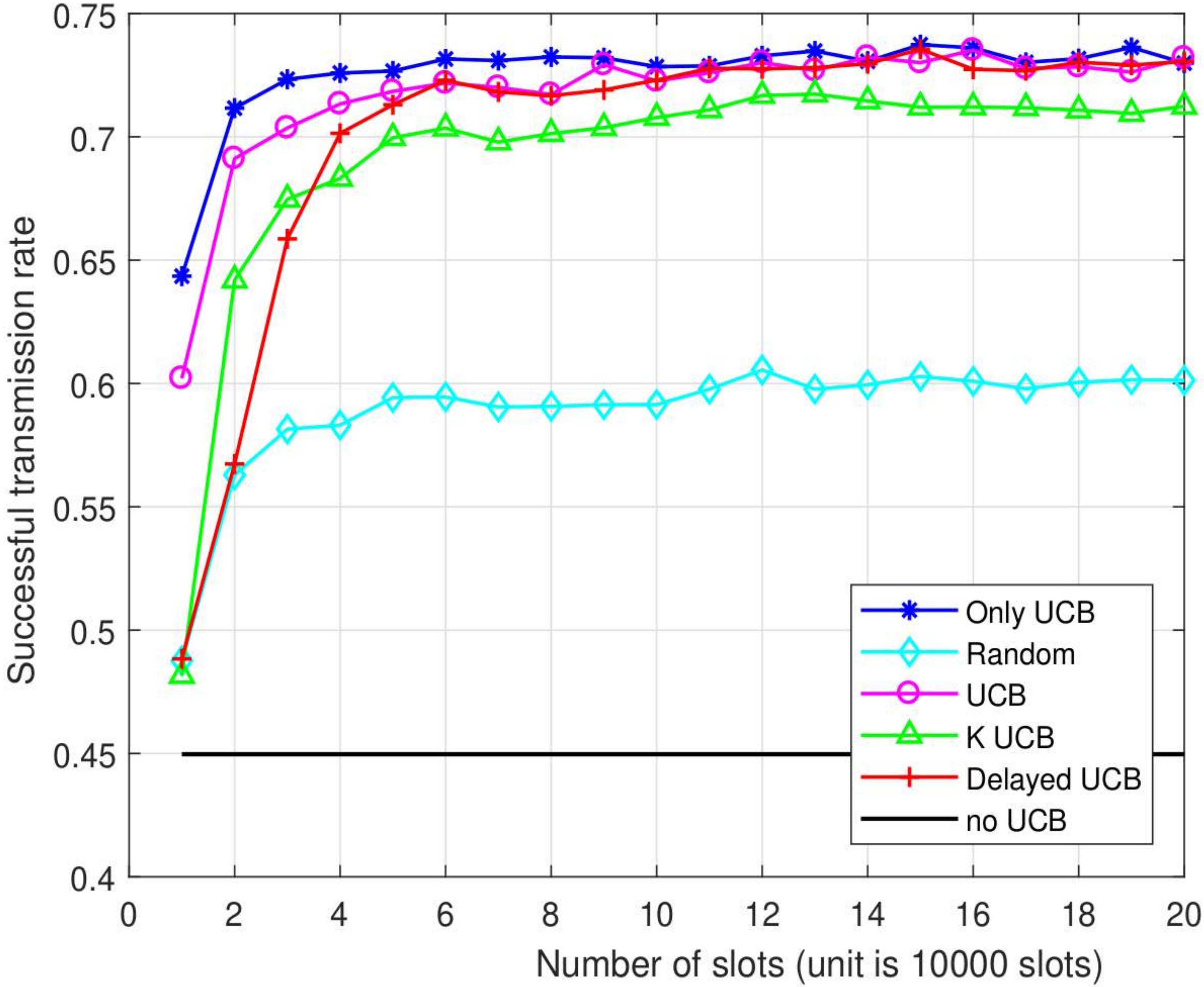

0.88
0.86
0.84
0.82
0.8
0.78
0.76
0.74
0.72
0.7
0.68
Successful transmission rate
0
2
4
6
8
10
12
14
16
18
20
Number of slots (unit is 10000 slots)
Only UCB
Random
UCB
K UCB
Delayed UCB
no UCB

5.2. Second experiment

- Same parameters

Hypothesis

👉 Non uniform occupancy of the 4 channels:
they are occupied $40, 30, 20$ and 30% of times (by other IoT networks).



Successful transmission rate
Number of slots (unit is 10000 slots)
0.75
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0
2
4
6
8
10
12
14
16
18
20
Only UCB
Random
UCB
K UCB
Delayed UCB
no UCB

6. Summary (1/3)

Settings

1. For **IoT networks** based on a simple **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,
4. We focus on the packet retransmissions upon radio collision, by using 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.

More ?

↪ **See our paper:** `HAL.Inria.fr/hal-02049824` 👈

🙏 **Please ask questions !**

Or by email `Lilian.Besson @ CentraleSupelec.fr` **?**

Thanks for listening 👍 *!*