Catching the LoRa ADR bandit with a new Sheriff: J-LoRaNeS

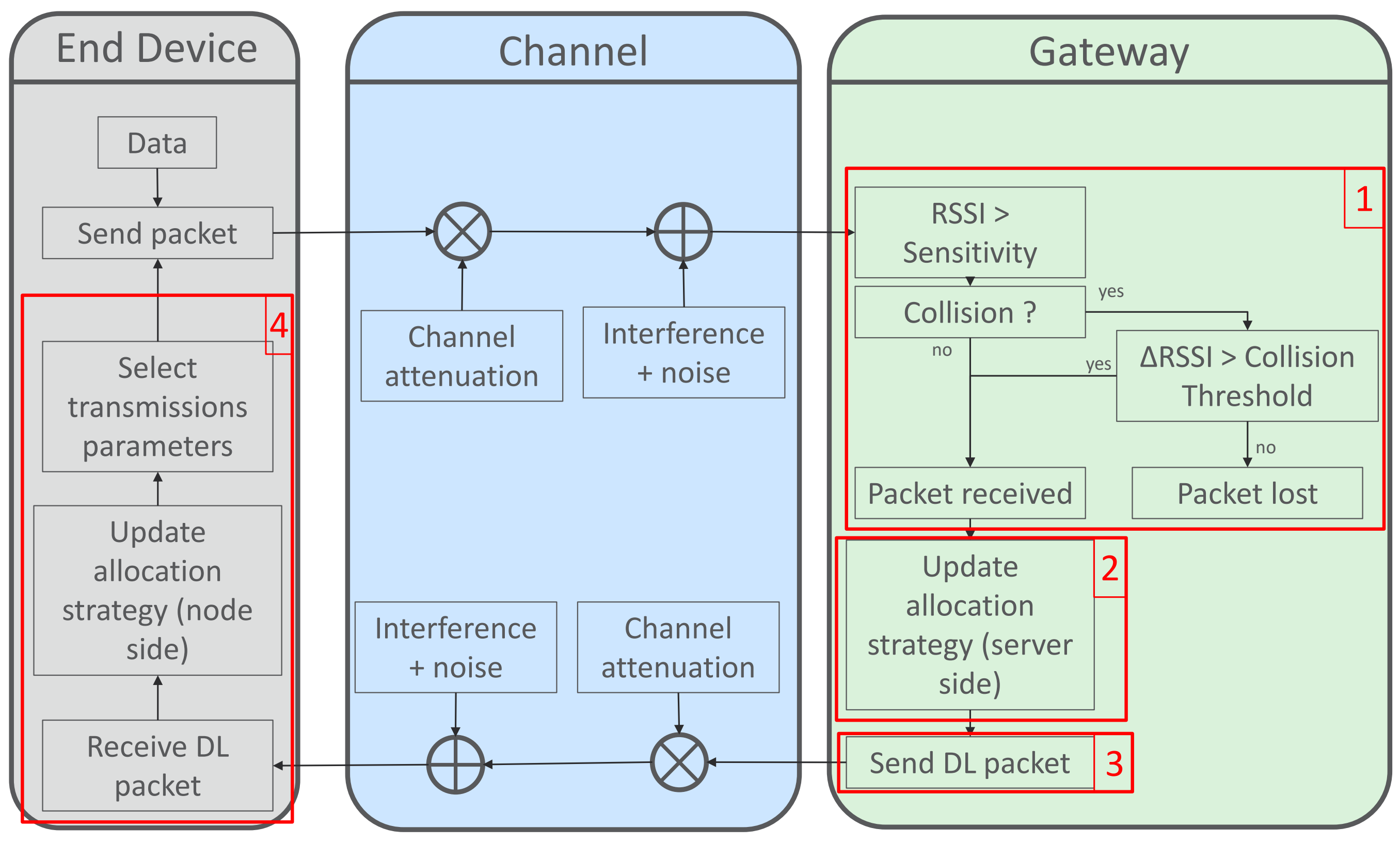
Context:

- Massive LPWAN deployment
- LoRa communication optimization
- LoRa Network simulation

Originality:

- We developed a LoRa network simulator highly customable with the Julia language
- Performance study of MAB-based Adaptive Data Rate algorithm with duty-cycle constraint (unreliable feedback)

J-LoRaNeS : A Julia based LoRa Network Simulator

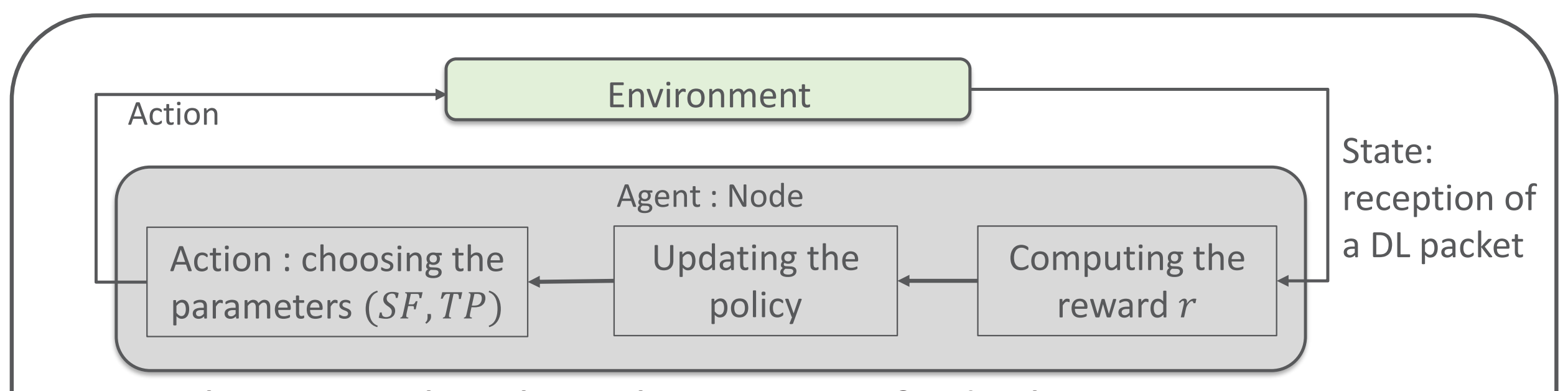

End Device
Data
Send packet
Select transmissions parameters
Update allocation strategy (node side)
Receive DL packet
4
Channel
Channel attenuation
Interference + noise
Interference + noise
Channel attenuation
Gateway
RSSI > Sensitivity
Collision ?
yes
no
yes
ΔRSSI > Collision Threshold
no
Packet received
Packet lost
1
Update allocation strategy (server side)
2
Send DL packet
3

We used the Julia language [1] to develop the simulator. This language was recently developed at the MIT, it is easy to use as Python, and since it is a compiled language, the code can be executed quickly. Moreover, Julia relies on multiple dispatch, which allows a huge flexibility for the simulator. Indeed multiple dispatch is a mechanism that will choose the right function behavior according to the input's type during the execution of the algorithm.

The simulated behavior of an End-Device(ED)/Gateway(GW) interaction is as follows:

1. The ED sends a packet, it needs to have a Received Signal Strength Indicator (RSSI) higher than a sensitivity threshold and a Signal to Interference Ratio threshold [3] to be received by GWs
2. The server updates the ADR mechanism on his side, and chooses a spreading factor (SF) and Transmit Power (TP) values for the ED next communication. Those values are sent to the ED through a Downlink (DL)message.
3. A DL packet is sent only if a GW is available and if it has received the corresponding uplink (UL).
4. The ED updates the policy and selects the new communication parameters for the next uplink message, even if the downlink message has not been received.

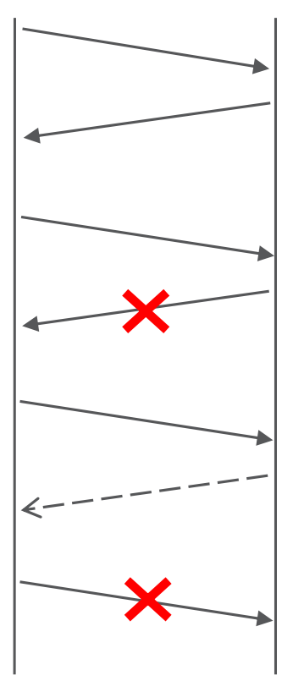
Multi-armed Bandit algorithm for parameters selection


Environment
Action
State: reception of a DL packet
Agent : Node
Action : choosing the parameters (SF, TP)
Updating the policy
Computing the reward r

- Exploration: the algorithm tries to find a better arm
- Exploitation: the algorithm uses the best arm, to his knowledge

Application of MAB to LoRa

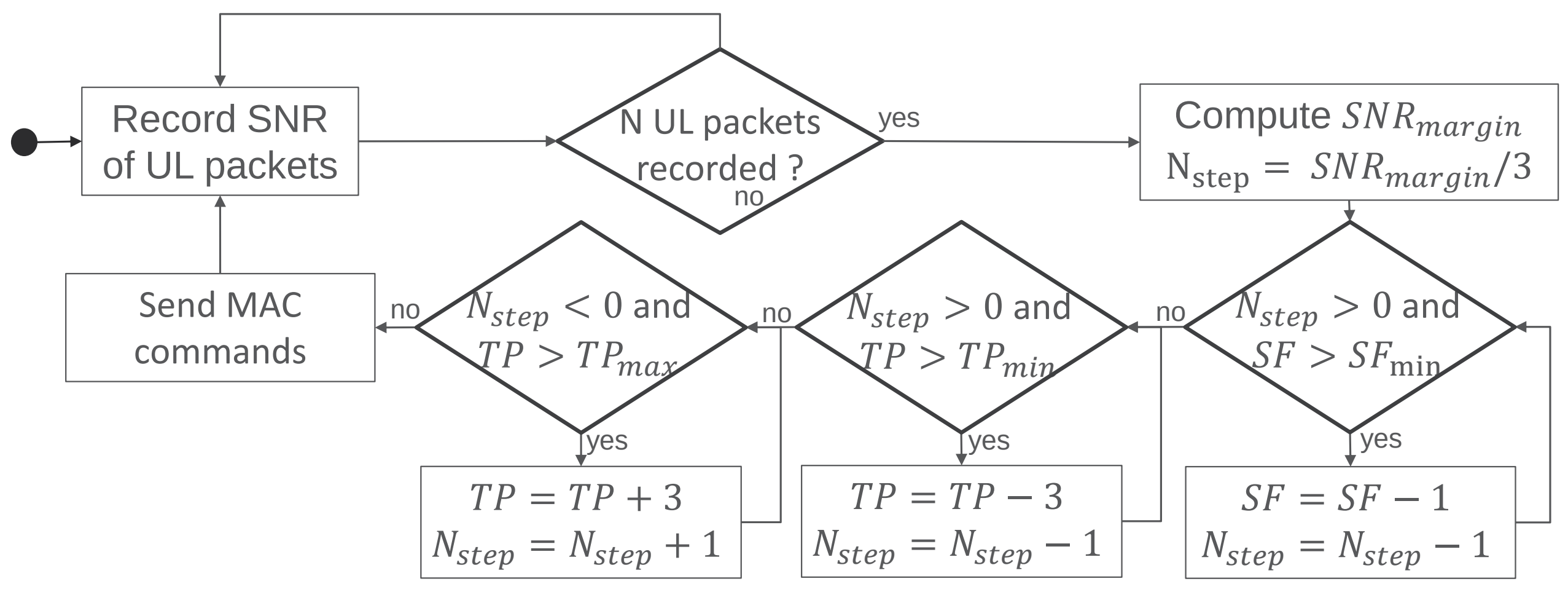
The study [2] demonstrates several flaws in the LoRaWAN ADR algorithm. Several propositions of enhancement have been made, and some of them are based on Multi-Armed Bandit (MAB) algorithm. But, those enhancements have an issue: they do not respect the duty-cycle of 1% or 10% imposed by regulation. The respect of duty-cycle can lead to bad arm performance estimation due to the absence of feedback. The question is then: **what happens to the performance of MAB-based ADR algorithms when the duty-cycle is properly respected?**



1. UL is received by the GW. After UL reception a DL is sent and received. Good arm performance estimation
2. UL is received by the GW. After UL reception a DL is sent and lost. Bad arm performance estimation
3. **UL is received by the GW. No DL sent after UL reception.** Bad arm performance estimation
4. UL is lost due to channel or collision. Good arm performance estimation

Possible interaction between ED and GW

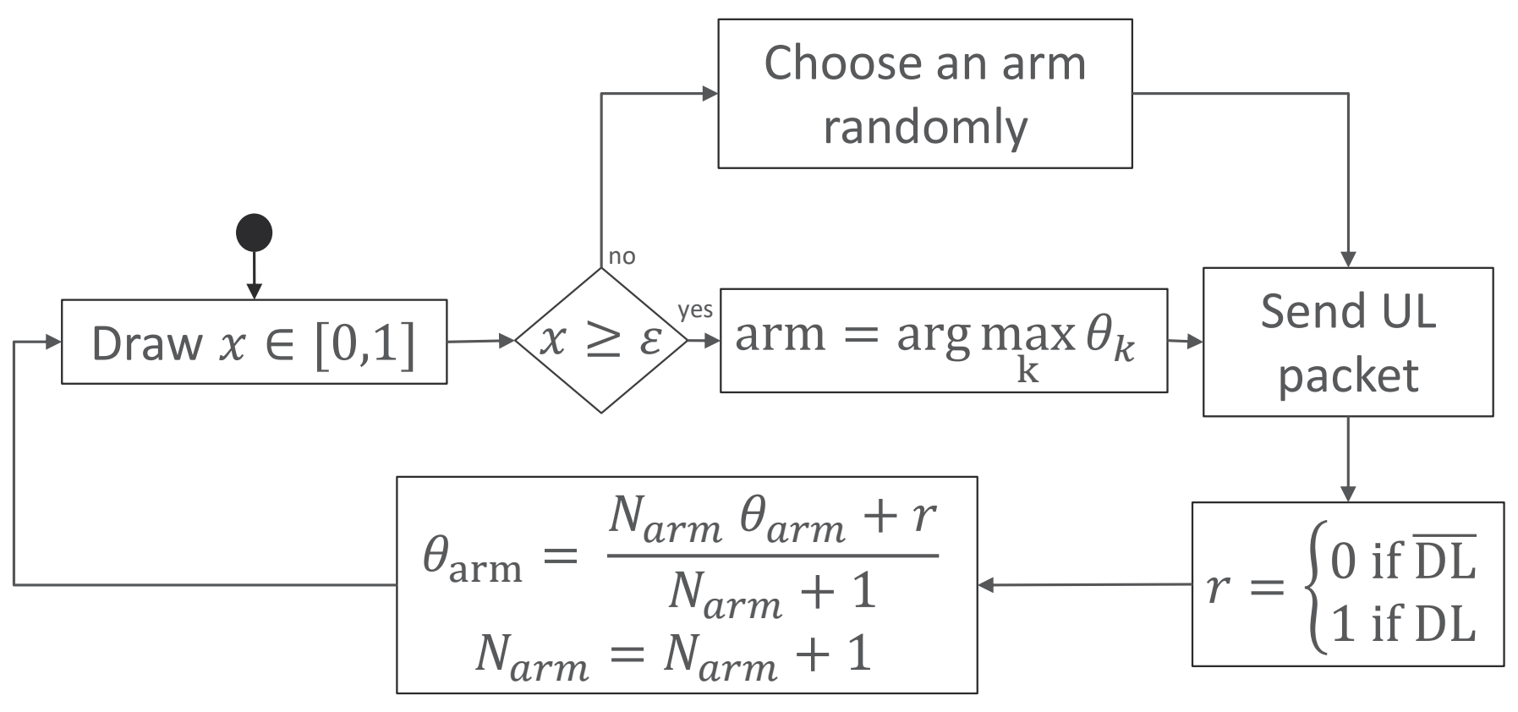
LoRaWAN ADR algorithm [4]


Record SNR of UL packets
N UL packets recorded ?
yes
no
Compute SNR_{margin}
$\mathrm{N_{step}} = SNR_{margin}/3$
Send MAC commands
no
$N_{step} < 0$ and $TP > TP_{max}$
yes
$TP = TP + 3$
$N_{step} = N_{step} + 1$
no
$N_{step} > 0$ and $TP > TP_{min}$
yes
$TP = TP - 3$
$N_{step} = N_{step} - 1$
no
$N_{step} > 0$ and $SF > SF_{\min}$
yes
$SF = SF - 1$
$N_{step} = N_{step} - 1$

The ADR is split into 2 algorithms with different objectives:

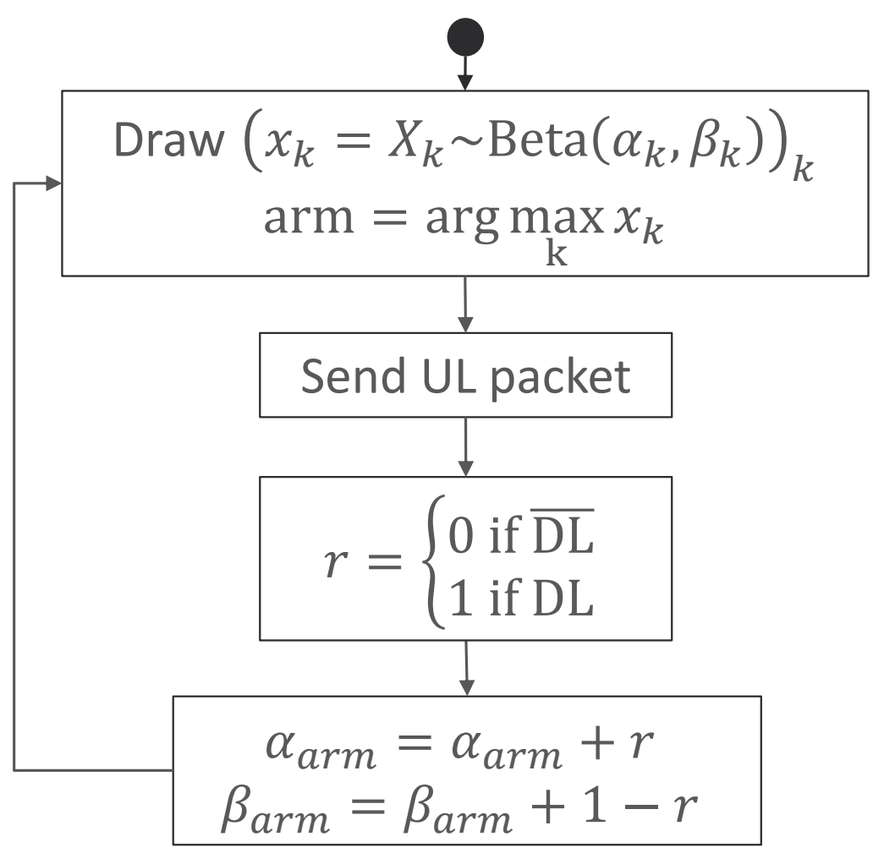
- Algorithm on server (presented above): make the communication as energy efficient as possible.
- Algorithm on node: reach a GW whatever the energy cost.

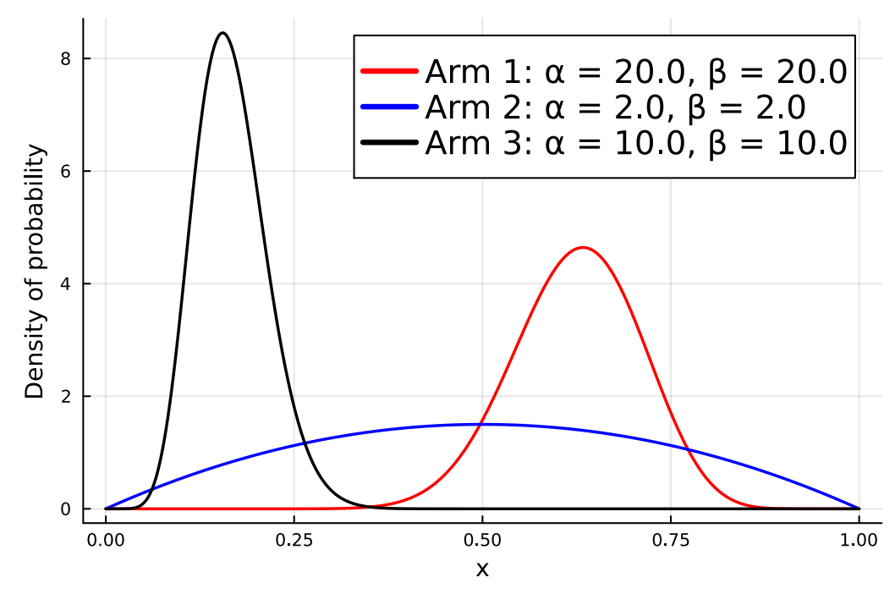
ε-Greedy ADR algorithm [5]


Choose an arm randomly
Draw $x \in [0,1]$
no
$x \geq \varepsilon$
yes
$\mathrm{arm} = \arg\max_{k} \theta_k$
Send UL packet
$\theta_{\mathrm{arm}} = \frac{N_{arm}\,\theta_{arm} + r}{N_{arm} + 1}$
$N_{arm} = N_{arm} + 1$
$r = \begin{cases} 0 \text{ if } \overline{\mathrm{DL}} \\ 1 \text{ if DL} \end{cases}$

The exploration probability ε can be fixed or dynamic. Here $\varepsilon = K / \sum_{k=1}^{K} N_k$, to ensure high exploration at the beginning, and high exploitation after a certain time.

Thomson-Sampling ADR algorithm [6]

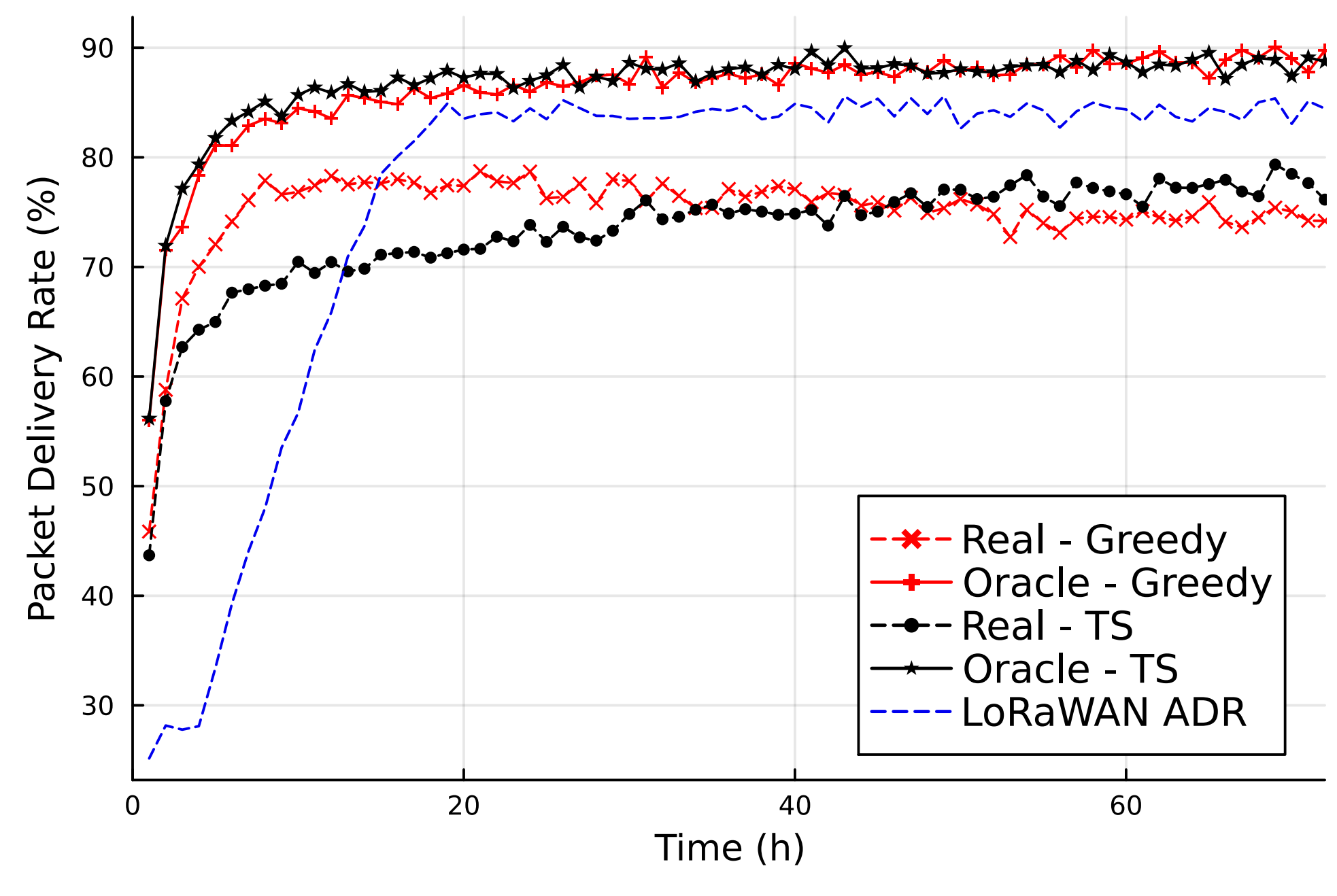

Draw $\left(x_k = X_k \sim \mathrm{Beta}(\alpha_k, \beta_k)\right)_k$
$\mathrm{arm} = \arg\max_{k} x_k$
Send UL packet
$r = \begin{cases} 0 \text{ if } \overline{\mathrm{DL}} \\ 1 \text{ if DL} \end{cases}$
$\alpha_{arm} = \alpha_{arm} + r$
$\beta_{arm} = \beta_{arm} + 1 - r$



Beta distribution example

The random variables allows to select the arms with high average rewards or the ones that are worth exploring, *i.e.*, with a high variance and rather high average.

ADR performance comparison



- Fast convergence time for Oracle algorithm, with higher performance than LoRaWAN ADR.
- But high penalty on performance for real cases, *i.e.,* respecting the duty-cycle at GW level.

Oracle: In this configuration the GW does not respect the duty-cycle.

Real: In this configuration the duty-cycle is respected by the GW.

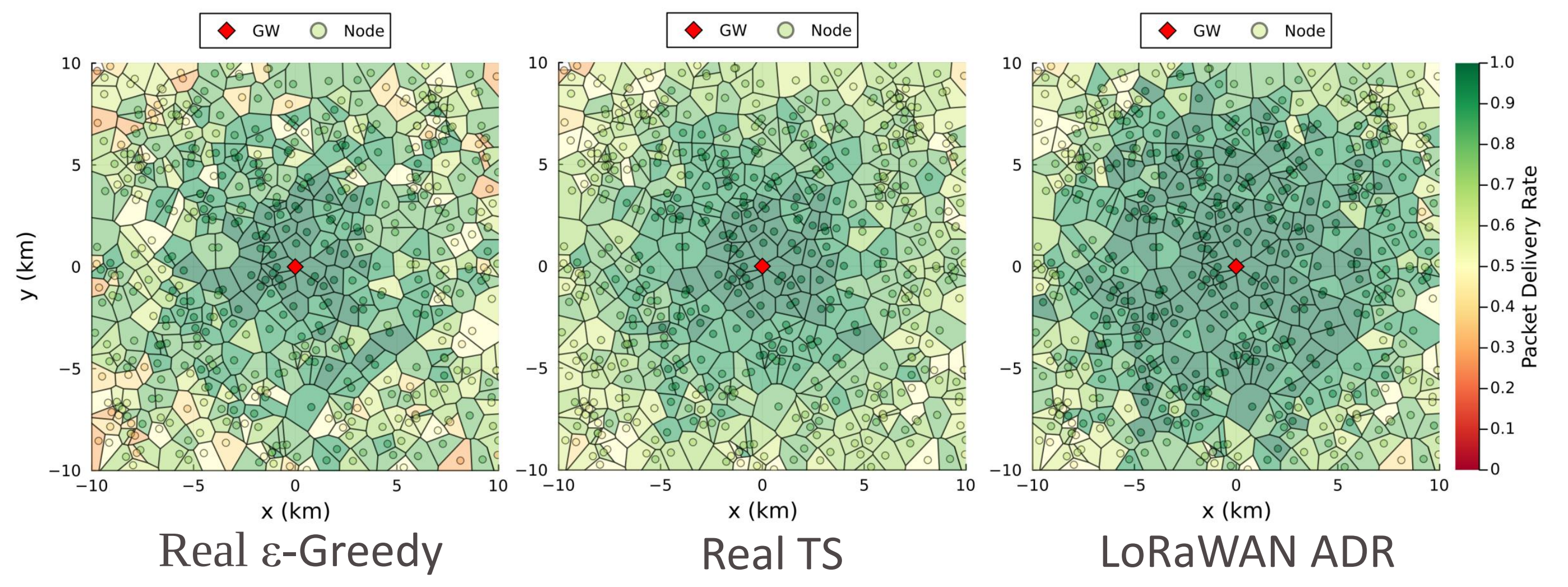
Simulation setup: 500 EDs, 1 GWs, 1 packet every 10 mins on average.

A Voronoi tessellation has been added on the map to help with the visualization.

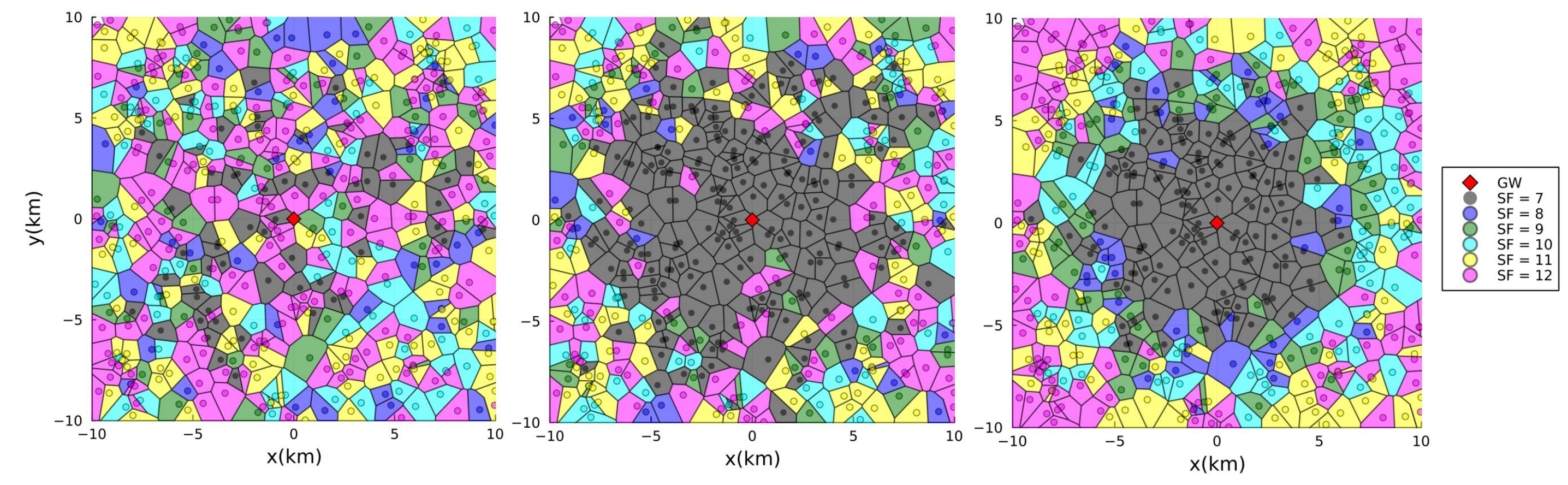
The PDR maps show that nodes far from the GW have bad communication performance for MAB-based algorithm. That lack of PDR is explained by a bad arm selection, as shown in most used SF maps. That bad selection is caused by the lack of feedback from the GW due to the duty-cycle. Nodes close to the GW are not affected because every configuration allows good performance in this case.

Network topology metrics

Packet Delivery Rate for each end-device of the network



Real ε-Greedy
Real TS
LoRaWAN ADR



Most used SF for each end-device of the network

Conclusions

- J-LoRaNeS is very flexible, and simulator behavior can be easily adapted
- The duty-cycle limitation can have a huge impact on the performance of MAB-based ADR algorithm, especially for dense networks.

Perspectives

- Find a MAB algorithm that can outperform the LoRaWAN ADR, even with duty-cycle limitation.
- Include energy in the reward computation to make the communication energy efficient

Jules COURJAULT[1]
Baptiste VRIGNEAU[1]
Olivier BERDER[1]
Claude GUICHAOUA[2]
Yvon LEGOFF[2]
[1]Univ Rennes, CNRS, IRISA
[2]CG-Wireless

Contacts

IRISA/GRANIT
6 rue de Kerampont – 22300 Lannion, France
firstname.lastname@irisa.fr
CG-Wireless
Le Moustoir, 29710 Plogastel St-Germain, France
firstname.lastname@cgwi.fr

References

[1] J. Bezanson & *al.*, "Julia: A Fresh Approach to Numerical Computing", *SIAM Review*, 2017
[2] S. Li & *al.* "How agile is the adaptive data rate mechanism of LoRaWAN?" *IEEE Global Communications conference (GLOBECOM)*, 2018
[3] D. Croce et *al.*, "Impact of LoRa imperfect orthogonality! Analysis of lin-level performance", *IEEE Communications Letters*, 2018
[4] LoRa Alliance, "LoRaWAN v1.1 specifications", 2017. Available at https://lora-alliance.org/resource_hub/lorawan-specification-v1-1/
[5] R. S. Sutton & *al. Reinforcement Learning: An introduction*, MIT press, 2018
[6] D. J. Russo & *al.*, "A tutorial on Thomson-Sampling", *Foundations and Trends in Machine Learning*, 2018