T3.1: Automating IoT device configuration.

1st June 2021

Renzo E. NAVAS (renzo.navas@inria.fr)

WP3: Virtualization and automation of IoT access networks.

Goals of Meeting

- **Explain you what are we are doing in T3.1**
- If possible, identify where the other partners (Acklio and Aguila?) can provide its expertise
    - (To Be Honest: we are relatively independent)

Recap T3.1: Goal

Description:

- Having a large number of IoT nodes to learn what parameters to use (time, power, spreading factor, etc.) for uploading data, and optimizing their global performance.
- Focus on mechanisms where IoT nodes make their own decisions (<u>decentralized</u>)
- But, end devices may have their strategy optimized globally by the orchestrator.

Goal:

- Propose and analyze the performance of machine learning algorithms that need few resources, like multi-armed bandit methods.
- **D3.1: Lightweight learning algorithms for massive IoT and analysis of their performance. (T15)**

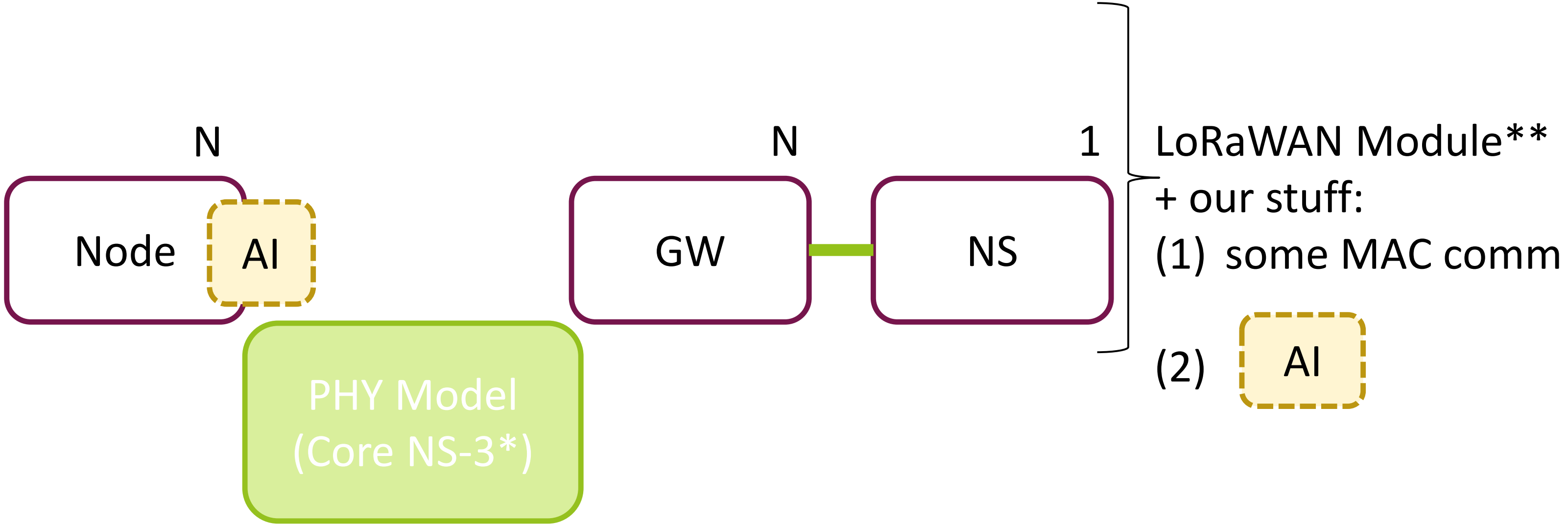
Recap T3.1: Roadmap

Baseline: Do “better” than LoRaWAN’s ADR –and SoA–, in massive IoT scenarios.

1. Related Work (Positioning, Survey. E.g. [1]) [Non-priority for D3.1]
2. **Proposal**: Reinforcement-Learning-based, particularly **Bandit Algorithms [2]**
3. **Evaluation**: We will use a realistic setting/evaluation scenario, NS-3 based [URL].
4. **Contribution:** A differentiating factor of our Bandit-based Algorithm(s) will be this applicability/evaluation in realistic LoRaWAN scenarios.
    - In the literature, proposals use strong hypotheses or simplified models.
    - However, evaluation/comparison against non-bandit proposals will be a challenge (i.e., implementation)

Outline

Software Components



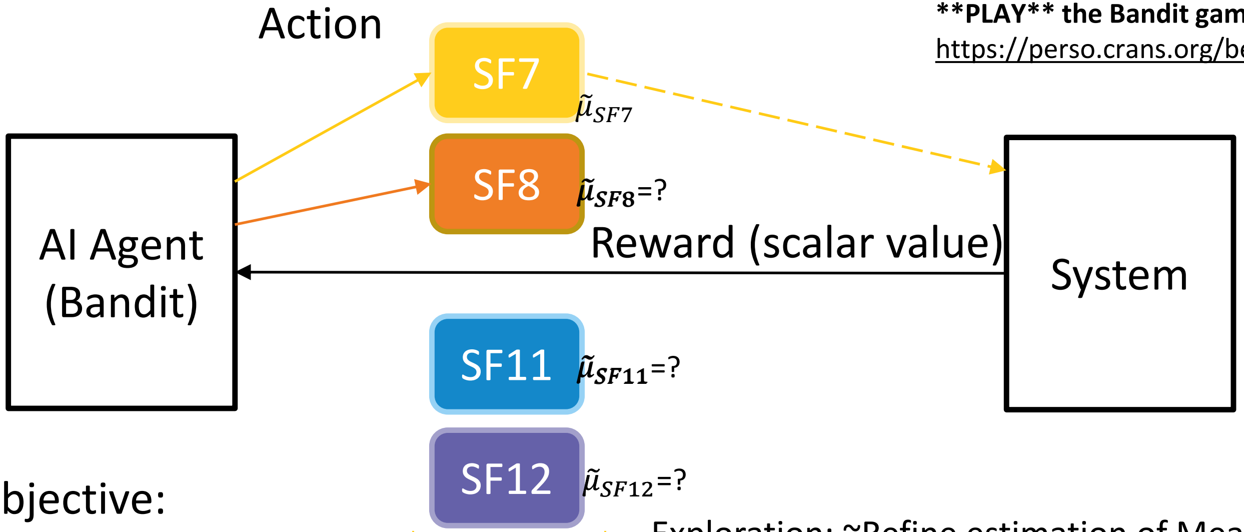
*Core NS-3: A Discrete Event Simulator (in C++)

**LoRaWAN NS3 Module: https://github.com/signetlabdei/lorawan

AI-kernel: https://github.com/Svalorzen/AI-Toolbox

Bandit: Playing a game

****PLAY** the Bandit game for yourself! (Nice, Lilian B!):**
https://perso.crans.org/besson/phd/MAB_interactive_demo/



k Arms

Objective:
Maximize the
long-run reward
(horizon N)

Exploration: ~Refine estimation of Mean (μ) (and Variance!) of Arms

Exploitation: ~Use the arm with current max mean reward.

(most bandit techniques mix exploration with exploitation)

A bit of Detail (ADR Bandit Agent)

```
// @brief A class that implements a Bandit RL Agent targeted at Adaptive Data Rate LoRaWAN End Nodes
class AdrBanditAgent : public Object
{
public:
   /*…*/

   int  ChooseArm ();
   //int  chooseArmFromPolicyN(int policyNumber); If we have more than one policy! :O, all learn from shared experience

   void UpdateReward (int armNumber, double reward);

protected:
     Experience      m_experience;
     PolicyInterface * m_aiPolicy; // We can have more than one ML-Policy ☺, all can learn from the same Experience.
};
```

First Assumptions AI

- 1 Bandit Arm == 1 Spreading Factor
  - (Frequency will be uniformly random, and power constant)
  - 1 arm generalizes to anything (phy/mac params), but I want to keep total number of arms low (at least for now)

- Working on non-mobility Use Case (for now).
  - Mobility? Whatever we learn here can be adapted for mobility use cases ("first we learn to walk, then to fly")

- Horizon infinite, BUT we want to converge relatively "fast" (i.e., Better tan ADR)
  - In practical terms, we can assume we want to **converge for ~64 uplink packets**.
  - (Mobility UC will be equivalent to a finite horizon, with active triggering of a new learning phase/movement)

Adding Bandits to Sim Model - I

- A first working implem. of "Thompson Sampling" (TS) Bandit Policy
  - New NS-3 Node: "Class A End Device LoRAWAN Mac Bandit"
  - "Bandit" is an interface: use of any bandit is trivial.
    - Also, can be extended for more-than-bandits (e.g., Markov Decision Processes), with a system state var.
    - More-than-bandits is OK in sim, but unlikely on IoT nodes ☺ (not practical usability)
  - Bandits C++ implementation from : https://github.com/Svalorzen/AI-Toolbox

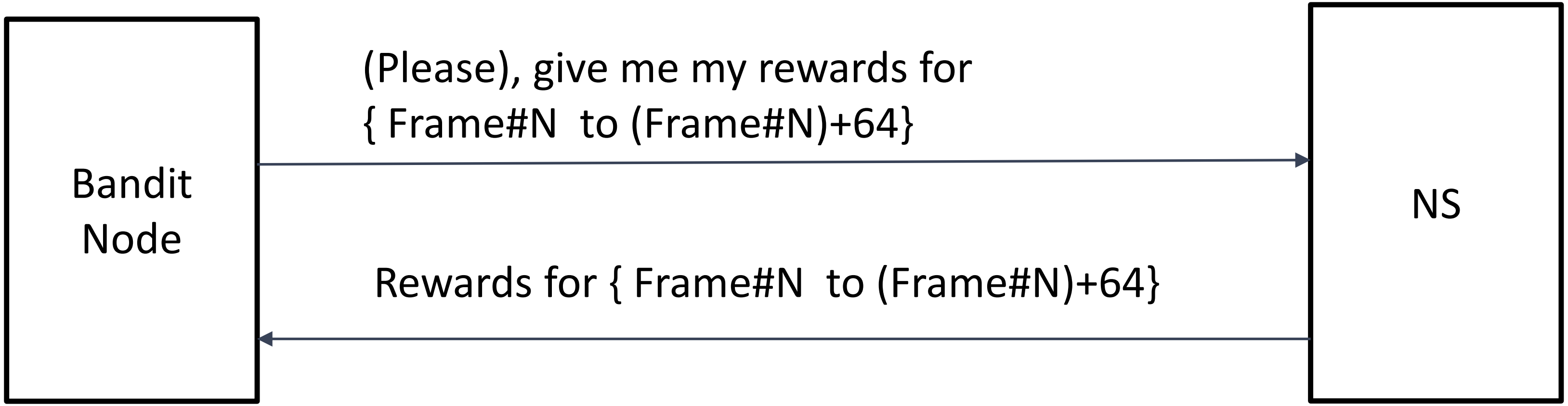
Inria

Adding Bandits to Sim Model - II

- The (my) current challenge is the "Bootstraping" of the policy (i.e., the first steps of the learning) → We want to converge "fast" (~<64 msgs)
  - TS (and other policies, I think) needs **two** samples **per arm** to start (to have a **mean** and a **variance**).
  - Heuristically, we can do whatever we want to help boostraping (e.g., assume a priori values of mean and var.)
  - *(My problem…)* We know it is a waste of time to sample randomly/blindly all the arms in this bootstrapping phase. Because they are **not** independent in our LoRa use case… *we know relevant things* :)
  - *Another big challenge*: our feedback/reward is not immediate (DownLink is Expensive, and degenerates the medium performance); but, we can assume that in the "boostraping phase" we *actively request DLs to help converge fast* (the rate of request will be adaptive).

Delayed Feedback/Reward

- Remember, the Feedback message (either request or response) can be lost!
  - Ergo it can **not** be relative (e.g., "Give me last N packets rewards"): needs an **absolute reference** (i.e., Frame#)



****Updates rewards for last 64 actions****

Wrapping Up:

- A Functional TS Bandit in a LoRaWAN Simulation (Good!)
- But, early stages:
  - Refine the “boostrapping” (including **REWARD** definition –fn(PDR,Energy)–)
  - Need to (define &) implement the “delayed feedback” MAC command
  - **Scalability** (N nodes, and proper way to gather Statistics)

Discussion

- Acklio, AGUILA involvement? (If it suits you)
- Sharing the Code (Private GitLab)
- Any Other?

Merci !

renzo.navas@inria.fr

Bibliography

[1] Kufakunesu, Rachel, Gerhard P. Hancke, and Adnan M. Abu-Mahfouz. "A survey on Adaptive Data Rate optimization in LoRaWAN: Recent solutions and major challenges." Sensors 20.18 (2020): 5044.

[2] Lattimore, Tor, and Csaba Szepesvári. Bandit algorithms. Cambridge University Press, 2020. URL: https://tor-lattimore.com/downloads/book/book.pdf

Slides from Previous Meeting

Bandit Algorithms – Introduction I

SF9

Reward: packet delivered

SF7

Round	1	2	3	4	5	6	7	8	9	10	11	
SF9	0		1		0	0				1	?	PDR = 2/5
SF7		1		0			0	0	0		?	PDR = 1/5

Arms (k=2)

Exploration-phase

Exploitation?

- Strategy: shall the IoT node keep using SF9, ignoring SF7?
- Or we attribute the poorer PDR performance of SF7 to “bad luck” and try it a few more times? How many more times? (Exploration-Exploitation trade-off)
- … What if we redefine the (negative) **reward** as the energy spent to deliver a packet?

Bandit Algorithms – Introduction II

- Goal: Maximize the cumulative reward, for a horizon of N “rounds” (could be infinite).
  - The definition of “reward” is fundamental (E.g., plain PDR vs Energy-aware PDR).

- In the basic bandit setting:
  - **Context-agnostic**: Arms’ rewards are independent (a reward for one arm does not give information about other arms’. E.g., a packet delivered with SF7 does not mean it also would have been delivered with SF9).
  - **Context-agnostic**: The learning agent **only** interacts with the system by “pulling” an arm and observing the empirical reward.
  - **Perfect Monitoring**: Empirical-reward observation (feedback) is immediate (… or almost).
  - **Stationary:** The underlying environment does not change (E.g., An arm’s reward’s “behavior” is always the same)

Bandit Algorithms – A realistic setting

- In our realistic setting:
  - **Contextual Bandits** → We can use contextual information (E.g., use DL for SNR stats, SFs are not-independent)
  - **Partial Monitoring** → We do not have perfect monitoring (E.g., DownLink is expensive)
  - **Non-stationary** → The environment, most likely, is not stationary.

- Defining an analytical model for this setting is not trivial, and theoretical bandit solutions does not exist (to the best of my knowledge).
- Our proposal(s) will use bandit solutions at their core, but probably will be mixed with **some heuristics**.
- Thanks to a realistic ns-3 evaluation/simulation environment, we will have strong **statistical guarantees** about their performance (and will compare with some SoA, including vanilla ADR). TBD: Obtain some theoretical guarantees.

LoRaWAN Evaluation: NS-3

- NS-3 LoRaWAN module
  - GIT : https://github.com/signetlabdei/lorawan and Documentation.
  - Current version has an implementation of LoRaWAN's ADR.
  - Does not implement: Class-B Nodes, Frame Counters (but Lost packets are traced in-software).
  - Energy-measurement module could be implemented w/reasonable effort.
  - NS-3 module is highly customizable: E.g., stack of path loss models (Shadowing, Buildings..).