Evaluating the energy consumption of adaptation tasks for a CPS in the Arctic Tundra

Antoine Omond*†, Issam Rais*, Hélène Coullon†
*Department of Computer Science, UiT The Arctic University of Norway, Tromsø, Norway
†IMT Atlantique, Nantes Université, École Centrale Nantes, CNRS, INRIA,
LS2N, UMR 6004, F-44000 Nantes, France
Corresponding authors: antoine.s.omond@uit.no, issam.rais@uit.no, helene.coullon@imt-atlantique.fr

***Abstract*—Cyber-Physical Systems deployed in scarce resource environments like the Arctic Tundra (AT) face extreme conditions. Nodes deployed in such environments have to carefully manage a limited energy budget, forcing them to alternate long sleeping and brief uptime periods. During uptimes, nodes can collaborate for data exchanges or computations by providing services to other nodes. Deploying or updating such services on nodes requires coordination to prevent failures (*e.g.*, sending new/updated API, waiting for service activation/deactivation, etc.). In a CPS with short uptime periods, such coordination can be energy-consuming due to low opportunities for communications.**

This paper evaluates and studies nodes' energy consumption during deploy or update tasks coordination according to different CPS configurations (*i.e.*, number of nodes, uptime duration, radio technology, or relay node availability). Results show high energy consumption in scenarios where nodes wake up specifically to deploy/update. It is shown that it is beneficial to do adaptation tasks while overlapping with existing uptimes (*i.e.*, reserved for observation activities).

This paper also evaluates and studies how nodes' uptime duration and relay node availability influence energy consumption. Increasing uptime duration can reduce energy consumption, up to 12%. Using an available relay node for communication reduces the energy consumption by 47% to 99%.

***Index Terms*—CPS, Deployment, Update, Coordination, Tundra, Energy Consumption**

I. Introduction

The Arctic Tundra (AT) is one of the most sensitive environments to climate change. Presently, much less than 1% of the AT is monitored. Therefore, to accurately study the impact of climate change on the AT, larger observations are needed.

The AT is a very large, hard-to-reach, and potentially dangerous ecosystem. The Distributed Arctic Observatory (DAO)[1] project proposes an observatory CPS mainly composed of Observation Nodes (ON) deployed in the AT. ONs are small devices monitoring the environment through physical instruments (*e.g.*, sensors) and running small computations. ONs can also collaborate for observations when located close to each other through local or temporary network connections.

The AT environment imposes extreme constraints on ONs, notably due to its lack of infrastructure (*e.g.*, power grid, network, and roads to physically access the area). ONs are most of the time isolated. Due to harsh weather and the absence of eligible harvesting mechanisms, providing a regular energy supply is not possible. Thus, ONs are on a limited energy budget, forcing them to sleep most of the time to extend their lifetimes. Each ON has its own uptime schedule and wake-up for activity for very short periods. Uptime schedules are not synchronized and rarely overlap, leaving few opportunities for communication.

In the DAO-CPS, ONs collaborate for observations by providing services to other ONs (*e.g.*, analysis, data aggregation). During their lifetimes, ONs have to adapt their service configurations (*e.g.*, deploying new services, updating existing ones) from scientists' requests, or from external events happening in the CPS or in the AT. Due to ONs relying on these services, coordination is required to prevent failures. However, in most cases, ONs cannot rely on a central authority to handle the coordination. Due to scarce connectivity between ONs, such coordination can be energy-consuming as coordination can take a long time to converge.

In the DAO, few CPS configurations may influence the coordination's energy consumption. ONs uptime duration, while being short, may vary [1]. Uptimes with a longer duration lead to potential larger overlaps between ONs, thus faster coordination. However, longer uptimes can also lead to larger energy consumption, leading to a trade-off. Few nodes in the DAO-CPS have larger energy budget than ONs. These nodes, called Relay Node (RN), could be leveraged to be synchronized with ONs from a specific neighborhood, to ease communications.

Considering non-synchronized sleeping ONs trying to deploy/update services, the contributions of the paper are an evaluation and a study of:

- the impact of ONs uptime duration on the energy

[1] https://site.uit.no/dao/

consumption;
- the impact on energy consumption of using an available RN for communications;
- the energy consumed by communication for coordination when using two different radio technologies, LoRa or NB-IoT;

The paper is structured as follows: Section II presents a description of the use-case, Section III describes the experimental setup, Section IV presents and discusses results, Section V presents the related work, and finally Section VI concludes this paper.

II. Use-case description

Figure 1 is an overview of the DAO-CPS. ONs are equipped with small single-board computers (e.g, Raspberry Pi), network capabilities (*e.g.*, LPWAN radio technologies such as LoRa or NB-IoT), and sensors (e.g optical and proximity cameras, temperature) [2]. This allows advanced capabilities on ONs like computing, collaboration, and service deploy/update. While ONs are responsible for observations and computations, RNs are notably used to help ONs communicate. RNs are also under a limited energy budget but equipped with more powerful batteries, making them more likely to be reachable by ONs. However, realistically such nodes might not have a full availability or be present in all neighborhoods. This paper makes the most favorable assumption by having an available and reachable RN by all ONs in the studied neighborhood.

Our use-case is inspired by an existing CPS deployment in the AT [2]. Our use-case considers n ONs hosting *measurement* services, sending observations to an ON hosting an *aggregation* service. The *aggregation* ON needs outputs from all *measurement* ONs to be functional. Thus, whenever changes happen either in the *aggregation* ON or *measurement* ONs (*e.g.*, type of measurement changed, service has been interrupted), ONs need to share data and coordinate their changes. Two coordination cases are studied: (1) initially *deploy* all services on ONs and (2) *update* all services on ONs.

Deploy: It takes the form of two synchronization barriers. First, the *aggregation* ON fetches configurations from all *measurement* ONs, to do its calibration and installation. Second, it waits for confirmation that all *measurement* ONs are actively sensing and performing observations. It then starts to listen and process observations, shared by *measurement* ONs.

Update: The *aggregation* ON and each *measurement* ON notify each other of their interruption/restart. Before updating, *measurement* ONs wait for confirmation that the *aggregation* ON stopped listening for observations. Then, the *aggregation* ON waits for confirmation that each *measurement* ON has been updated before doing its own update. In the context of the DAO-CPS, we consider that updating a *measurement* ON usually takes a few seconds to complete. Thus, the first and second notifications between each *measurement* ON and the *aggregation* ON can happen in the same overlap. For this reason, *update* is considered faster than *deploy*, in most cases.

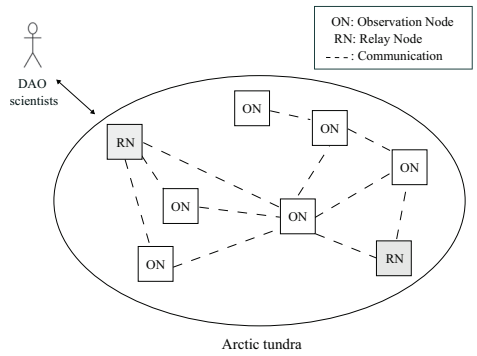

Fig. 1: DAO-CPS system in the Arctic Tundra. ONs are most of the time isolated from external communications. RNs can be available to few ONs.

ONs are able to run multiple services at the same time. During *deploy*/*update*, ONs can have other sensing/observing/computing activities. Having multiple activities allows multiple services to benefit from a single ON's uptime. The least and most favorable cases are considered. The least favorable case is waking up ONs specifically for *deploy*/*update*, where the uptime is dedicated only to coordination. The most favorable case is having other activities running on ONs besides *deploy*/*update*, taking advantage of existing uptimes.

In the DAO-CPS, *deploy*/*update* are usually low-priority tasks, as the energy budget of ONs should be reserved for observations and computations. Thus, synchronizing the wake-up schedules of ONs to ease communications during *deploy*/*update* is not considered in this paper.

III. Experimental setup

Experiments were conducted on ESDS [3], a flow-level network simulator validated and publicly available[2]. Simulation provides flexibility and saves time compared to setting-up and executing on a real infrastructure. The equivalent of several years of real execution time was reduced to a few hours of simulation. Simulation allows for easier reproducibility, as executions and results are not bound to a specific hardware platform.

In ESDS, an API is provided to simulate ONs' activity and to compute the energy consumption of each ON. The CPS is simulated as a network of ONs. For each ON, the sleeping and uptime periods, the

[2]https://gitlab.com/manzerbredes/esds

execution of *deploy*/*update* tasks, and communications are simulated.

A. Use-case simulation

Each ON can go through 4 states: *off*, *idle*, *stress* and *pull*. In the *off* state, the ON sleeps. It cannot receive or send messages or execute *deploy*/*update* tasks. In the *idle* state, the ON is awake. It can receive messages but doesn't execute *deploy*/*update* tasks and is not sending messages. In the *stress* state, the ON is awake and executes *deploy*/*update* tasks. Finally, in the *pull* state, the ON is awake but requires coordination data for one or multiple *deploy*/*update* tasks.

In the *pull* state, ONs request data once per second. This high frequency is motivated by the uptime characteristics of our use-case. ONs wake-up once every hour for very short periods of time, leading to very few overlaps. Such high frequency minimizes the probability of missing an overlap. Here, coordination data is assumed to be a simple flag. Each request has a measured fixed size of 257 bytes, close to the lower bound of common request sizes[3].

In the *stress* state, only the load generated on the ON to execute *deploy*/*update* tasks is considered. The load varies depending on the task. In the simulations, the worst-case is considered: each *deploy*/*update* task fully stresses the ON. Each task is assigned to a random duration, following a lognormal distribution bounded between 1s and 30s, where low values are more represented. This interval is chosen according to [1] where the minimum ON uptime duration is 1min.

To compute the ON states and *deploy*/*update* duration, a dependency graph is created. Each arc represents a task, directed either toward the next task on the same ON, or toward a task on a remote ON. Computing a graph traversal gives the ONs states and their duration for each individual ON. Computing the longest path gives the total *deploy*/*update* duration. The graph creation is inspired by [4]. The ON states, their duration, and the *deploy*/*update* duration are given as input to ESDS. For each ON, ESDS simulates these states and gives the energy consumed.

The draw of tasks duration, uptime schedules, ONs states, and simulation results are available online[4].

B. Simulation parameters

Table I summarises parameters used to conduct simulations. *Deploy* and *update* are simulated. During simulations, ONs uptime duration is set either at 1 min, 2 min or 3 min. [1]. For coordinations, two types of communication are considered: either direct (*i.e.*, ON-to-ON) or using an available RN, as an intermediary. In the DAO-CPS, ONs neighborhood sizes can vary up to 29 ONs [5]. To cover most cases, simulations are run using 1 *aggregation* ON and 5, 15, or 30 *measurement* ONs. Finally, two suitable radio technologies for the DAO-CPS [1] are simulated: LoRa and NB-IoT.

TABLE I: Summary of simulation parameters

Coordination name		{*deploy*, *update*}
Communication type		{direct, rn}
# of measurement ONs		{5, 10, 15, 20, 25, 30} [5]
Upt duration		{1min, 2min, 3min} [1]
Bandwidth (ltnc)	LoRa	50 kbps (0s) [1]
	NB-IoT	200 kbps (0s) [1]
Energy values	P_{idle}	1,339W [6]
	P_{stress}	2,697W [6]
	LoRa	+0.16W [1]
	NB-IoT	+0.65W [1]

The energy calibration is done from the energy consumption of a Raspberry Pi based ON, previously used in papers dealing with CPS deployment in the AT [5], [7]. The extreme values measured in [6] are used (1.339W for P_{idle}, and 2.697W for P_{stress}). When an ON is sleeping, its energy consumption is assumed to be null. Finally, the communication cost (send or receive) is calibrated from [1] (additional 0.16W for LoRa, 0.65W for NB-IoT).

C. Metrics

To get a representative set of results, 200 uptime schedules are generated. During each schedule, ONs wake-up randomly every hour. In our experiments, a scenario represents a combination of parameters. Each scenario is run on each uptime schedule, for a total of 200 runs per scenario.

The accumulated energy consumed by all ONs under each scenario and each uptime schedule is composed of static and dynamic energy consumption. The static energy consumption is the energy passively consumed by ONs during uptime. The dynamic energy consumption is the energy consumed for the execution and coordination of *deploy*/*update* tasks. For 200 runs, the means of the accumulated static, dynamic and total energy consumed by all ONs under a scenario s are denoted $\overline{eStatic}(s)$, $\overline{eDynamic}(s)$ and $\overline{e}(s)$, respectively. $\overline{e}(s)$ is given by

$$\overline{e}(s) = \overline{eStatic}(s) + \overline{eDynamic}(s) \quad (1)$$

Distinguishing between $\overline{e}$ and $\overline{eDynamic}$ allows to consider when ONs wake-up specifically for *deploy*/*update* and when ONs take advantage of existing uptimes. When ONs wake-up specifically for *deploy*/*update*, $\overline{e}$ is considered. When ONs take advantage of existing uptimes, only $\overline{eDynamic}$ is considered.

[3]https://www.chromium.org/spdy/spdy-whitepaper/

[4]https://github.com/aomond-imt/reconfiguration-esds/releases/tag/greencom_2023

$\overline{eComms}(s)$, included in $\overline{eDynamic}$, represents the mean of the accumulated energy consumed by communications for 200 runs under a scenario s. This allows to compare the coordination's energy consumption when using LoRa or NB-IoT. Finally, $\bar{t}(s)$ represents the mean of the duration of *deploy*/*update*, for 200 runs under a scenario s.

For each metric m, the percentage of variation $\%\Delta m(s_1, s_2)$ quantifies the variation of m from scenario s_1 to s_2.

$$\%\Delta m(s_1, s_2) = \frac{m(s_1) - m(s_2)}{m(s_1)} * 100, \qquad (2)$$

where m can be either $\bar{e}$, $\overline{eDynamic}$, $\overline{eComms}$ or $\bar{t}$.

The percentage of variation is used to study the impact of varying simulation parameters on energy consumption and coordination duration.

IV. Evaluation

This section presents the energy consumed by ONs for each scenario and compares results. Scenarios are created from the Cartesian product of simulation parameters, described in Table I: coordination name, communication type, number of *measurement* ONs, uptime duration, and simulated radio technology.

Results show that, under most scenarios, waking up ONs specifically for the *deploy*/*update* coordination implies non-realistic energy consumption. Enabling faster coordination convergence, by increasing ONs uptime duration, has a limited impact on energy consumption reduction. Having an RN is always favorable with regard to energy consumption. However, in the AT, the availability of an RN is not always guaranteed.

When ONs run *deploy*/*update* tasks along with their observation activities, only the dynamic energy used for adaptation tasks is taken into account. Increasing the uptime duration only increases energy consumption. Having an RN is also always favorable with regard to energy consumption. Finally, LoRa has a lower consumption than NB-IoT, in all scenarios.

Previous observations are detailed in following sections. Only the most relevant and significant elements are extracted and analyzed.

A. Energy consumed when ONs wake up specifically for deploy and update

Table II presents $\bar{e}$ for *deploy* and *update* according to the number of *measurement* ONs, the uptime duration and the type of communication (*i.e.*, direct, rn). Note that $\overline{eDynamic}$ is also depicted in the tables but is analyzed in Section IV-B.

For the smallest cluster size (5 *measurement* ONs), 1 min uptime duration and using direct communications, $\bar{e}$ is equal to 63,82 kJ for *deploy*, and to 39,18 kJ for *update*. For the biggest cluster size (30 *measurement* ONs), $\bar{e}$ reaches 572,02 kJ for *deploy*, and 350,34 kJ for *update*. These high energy consumption are expected due to the characteristics of our use-case. Scarce connectivity between ONs leads to a significant amount of time required for coordination convergence. This is illustrated by Table III, where values for $\bar{t}$ are shown according to the number of *measurement* ONs, the uptime duration and the type of communication. For 5 *measurement* ONs, 1 min uptime duration and using direct communications, *deploy* and *update* take in average 143,19 hours and 87,61 hours respectively. For 30 *measurement* ONs, *deploy* and *update* take in average 248,65 hours and 152,02 hours respectively.

Uptime duration: Modifying uptime duration leads to a trade-off between energy saved from faster convergence and energy spent at each uptime. Its impact on $\bar{e}$ is shown in Table II. For *deploy* and the smallest cluster size, increasing uptime duration from 1 to 3 min decreases $\bar{e}$ by 6,33% (from 63,82 to 59,78 kJ). For the largest cluster size, increasing uptime duration from 1 to 2 min slightly decreases $\bar{e}$ by 1,04% (from 572,02 to 566,06 kJ). When increasing uptime duration from 1 to 3 min, $\bar{e}$ slightly increases by 1,11% (from 572,02 to 578,35 kJ). For *update* and the smallest cluster size, increasing uptime duration from 1 to 3 min decreases $\bar{e}$ by 11,97% (from 39,18 to 34,49 kJ). For the highest cluster size, $\bar{e}$ decreases by 12,25% (from 350,34 to 307,41 kJ).

Increasing uptime duration allows for a strong reduction in $\bar{t}$. The reduction in coordination duration is shown in Table III. For *deploy* and the smallest cluster size, increasing uptime duration from 1 to 3 min decreases $\bar{t}$ by 68,78% (from 143,19 to 44,71 hours). For *update*, $\bar{t}$ decreases by 70,60% (from 87,61 to 25,76 hours). For *deploy* and the largest cluster size, $\bar{t}$ decreases by 66,31% (from 248,65 to 83,76 hours). For *update*, $\bar{t}$ decreases by 70,81% (from 152,02 to 44,38 hours).

Communications using an available RN: Table II shows results when using an available RN for communications. ONs energy consumption drastically decreases: for 1 min uptime duration and any cluster size, $\bar{e}$ decreases by more than 97% for *deploy*, and by more than 94% for *update*. These reductions are expected, as using an RN drastically reduces *deploy*/*update* duration: for 1 min uptime duration having an RN reduces $\bar{t}$ by more than 97% under any scenario (from Table III).

Finally, combining both uptime duration and RN leverages increases energy consumption under all scenarios (Table II). When using an RN for *deploy*, increasing uptime duration from 1 to 3 min for the smallest cluster size increases $\bar{e}$ from 1,39 to 3,40 kJ. For the largest cluster size, $\bar{e}$ increases from 6,91 to 16,50 kJ. Similar variations can be observed for *update*. For the smallest cluster size, increasing uptime

TABLE II: $\overline{e}$ and $\overline{eDynamic}$ values for *deploy*/*update* according to the number of *measurement* ONs, the uptime duration, and the type of communication. Standard deviations are shown in parentheses. $\%\Delta\overline{e}$ and $\%\Delta\overline{eDynamic}$ quantify the variation of $\overline{e}$ and $\overline{eDynamic}$, from scenarios using direct communications and scenarios using an available RN.

deploy						
5 *measurement* ONs						
	$\overline{e}$			$\overline{eDynamic}$		
	direct (kJ)	rn (kJ)	$\%\Delta\overline{e}$	direct (J)	rn (J)	$\%\Delta\overline{eDynamic}$
1min uptime	63,82 (24,18)	1,39 (0,31)	97,82	221,84 (31,29)	104,24 (4,83)	53,01
2min uptime	61,46 (24,68)	2,51 (0,62)	95,92	221,99 (32,49)	107,28 (6,90)	51,67
3min uptime	59,78 (23,64)	3,40 (0,92)	94,31	224,01 (33,38)	109,81 (10,37)	50,98
15 *measurement* ONs						
	$\overline{e}$			$\overline{eDynamic}$		
	direct (kJ)	rn (kJ)	$\%\Delta\overline{e}$	direct (J)	rn (J)	$\%\Delta\overline{eDynamic}$
1min uptime	245,94 (61,29)	3,68 (0,73)	98,50	575,59 (44,23)	255,52 (6,27)	55,61
2min uptime	248,08 (67,00)	6,61 (1,49)	97,34	623,47 (58,84)	265,23 (11,18)	57,46
3min uptime	240,56 (62,96)	9,31 (2,09)	96,13	658,32 (67,97)	273,76 (17,69)	58,42
30 *measurement* ONs						
	$\overline{e}$			$\overline{eDynamic}$		
	direct (kJ)	rn (kJ)	$\%\Delta\overline{e}$	direct (J)	rn (J)	$\%\Delta\overline{eDynamic}$
1min uptime	572,02 (123,12)	6,91 (1,29)	98,79	1081,37 (66,98)	450,39 (10,66)	58,35
2min uptime	566,06 (113,78)	11,82 (2,49)	97,91	1228,90 (80,45)	467,23 (18,88)	61,98
3min uptime	578,35 (134,04)	16,50 (3,63)	97,15	1381,37 (104,40)	483,82 (28,54)	64,98
update						
5 *measurement* ONs						
	$\overline{e}$			$\overline{eDynamic}$		
	direct (kJ)	rn (kJ)	$\%\Delta\overline{e}$	direct (J)	rn (J)	$\%\Delta\overline{eDynamic}$
1min uptime	39,18 (17,67)	2,12 (0,46)	94,59	222,38 (47,24)	111,13 (3,47)	50,03
2min uptime	35,29 (18,19)	3,90 (1,02)	88,95	222,48 (52,29)	121,88 (8,96)	45,22
3min uptime	34,49 (17,75)	5,87 (1,40)	82,98	219,72 (47,56)	136,31 (11,57)	37,96
15 *measurement* ONs						
	$\overline{e}$			$\overline{eDynamic}$		
	direct (kJ)	rn (kJ)	$\%\Delta\overline{e}$	direct (J)	rn (J)	$\%\Delta\overline{eDynamic}$
1min uptime	150,81 (44,89)	5,94 (0,95)	96,06	620,27 (87,62)	269,24 (9,53)	56,59
2min uptime	141,18 (43,27)	10,46 (2,20)	92,59	668,75 (92,83)	298,41 (26,61)	55,38
3min uptime	126,80 (49,71)	15,13 (2,99)	88,07	690,84 (116,43)	343,76 (36,19)	50,24
30 *measurement* ONs						
	$\overline{e}$			$\overline{eDynamic}$		
	direct (kJ)	rn (kJ)	$\%\Delta\overline{e}$	direct (J)	rn (J)	$\%\Delta\overline{eDynamic}$
1min uptime	350,34 (107,06)	11,11 (1,66)	96,83	1274,14 (141,42)	505,62 (19,83)	60,32
2min uptime	321,13 (83,48)	19,38 (3,66)	93,97	1476,30 (163,06)	568,08 (55,87)	61,52
3min uptime	307,41 (88,20)	26,98 (4,59)	91,22	1664,77 (177,83)	665,37 (81,07)	60,03

duration also increases $\overline{e}$ from 2,12 to 5,87 kJ. For the largest cluster size, $\overline{e}$ increases from 11,11 to 26,98 kJ. These variations are explained by the full availability of the RN for ONs. In such conditions, the impact of increasing uptime duration on the *deploy*/*update* coordination duration is minimized.

B. Energy consumed when ONs take advantage of existing uptimes for deploy and update

In this section, only $\overline{eDynamic}$ is considered, as the *deploy*/*update* coordination is considered to be a task running among others, on ONs. The RN is also included in this assumption, as ONs use RN not only for *deploy*/*update* but also for any type of collaboration. Table II shows that, for 1 min uptime duration, $\overline{eDynamic}$ for *deploy* is 221,84 J for the smallest cluster size, and 1081,37 J for the highest cluster size. For *update*, $\overline{eDynamic}$ is 222,38 J for the smallest cluster size, and 1274,14 J for the highest cluster size.

Uptime duration: When uptime duration increases, under most scenarios $\overline{eDynamic}$ increases, especially for large cluster sizes (Table II). For *deploy* and the smallest cluster size, increasing uptime duration from 1 to 3 min slightly increases $\overline{eDynamic}$ by 0,98% (from 221,84 to 224,01 J). For the largest cluster size, $\overline{eDynamic}$ increases by 27,74% (from 1081,37 to 1381,37 J). For *update* and the smallest cluster size, increasing uptime duration from 1 to 3 min slightly decreases $\overline{eDynamic}$ by 1,20% (from 222,38 to 219,72 J). For the largest cluster size, $\overline{eDynamic}$ increases

TABLE III: $\bar{t}$ values for *deploy*/*update* according to the number of *measurement* ONs, the uptime duration, and the type of communication. Standard deviations are shown in parentheses. $\%\Delta\bar{t}$ quantifies the variation of $\bar{t}$ from scenarios using direct communications and scenarios using an available RN.

deploy			
5 *measurement* ONs			
	direct (hours)	rn (hours)	$\%\Delta\bar{t}$
1min uptime	143,19 (54,37)	1,46 (0,34)	98,98
2min uptime	68,96 (27,71)	1,42 (0,38)	97,94
3min uptime	44,71 (17,70)	1,34 (0,41)	97,00
15 *measurement* ONs			
	direct (hours)	rn (hours)	$\%\Delta\bar{t}$
1min uptime	207,05 (51,67)	1,51 (0,31)	99,27
2min uptime	104,42 (28,25)	1,48 (0,33)	98,58
3min uptime	67,49 (17,69)	1,50 (0,34)	97,78
30 *measurement* ONs			
	direct (hours)	rn (hours)	$\%\Delta\bar{t}$
1min uptime	248,65 (53,60)	1,53 (0,30)	99,38
2min uptime	122,98 (24,76)	1,49 (0,31)	98,79
3min uptime	83,76 (19,46)	1,52 (0,34)	98,19

update			
5 *measurement* ONs			
	direct (hours)	rn (hours)	$\%\Delta\bar{t}$
1min uptime	87,61 (39,68)	2,30 (0,55)	97,37
2min uptime	39,47 (20,42)	2,24 (0,58)	94,32
3min uptime	25,76 (13,31)	2,30 (0,54)	91,07
15 *measurement* ONs			
	direct (hours)	rn (hours)	$\%\Delta\bar{t}$
1min uptime	126,70 (37,83)	2,50 (0,43)	98,03
2min uptime	59,27 (18,24)	2,39 (0,52)	95,97
3min uptime	35,44 (13,97)	2,43 (0,48)	93,14
30 *measurement* ONs			
	direct (hours)	rn (hours)	$\%\Delta\bar{t}$
1min uptime	152,02 (46,58)	2,55 (0,40)	98,32
2min uptime	69,58 (18,15)	2,49 (0,47)	96,42
3min uptime	44,38 (12,78)	2,50 (0,42)	94,37

by 30,66% (from 1274,14 to 1664,77 J). Increasing uptime duration leads to larger and more frequent overlaps between ONs. More overlaps leads to more ONs receiving communications, including non-intended transmissions. Receiving non-intended communications can add a significant overhead to $\overline{eDynamic}$.

Communications using an available RN: Using an available RN for communications allows further decreases in energy consumption (Table II). For *deploy*, 1 min uptime duration and the smallest cluster size, using an RN decreases $\overline{eDynamic}$ by 53,01% (from 221,84 to 104,24 J). For the largest cluster size, $\overline{eDynamic}$ decreases by 58,35% (from 1081,37 to 450,39 J). For *update*, 1 min uptime duration and the smallest cluster size, using an RN decreases $\overline{eDynamic}$ by 50,03% (from 222,38 to 111,13 J). For the largest cluster size, $\overline{eDynamic}$ is reduced by 60,32% (from 1274,14 to 505,62 J).

TABLE IV: $\overline{eComms}$ values for *deploy*/*update* according to the number of *measurement* ONs, the uptime duration, and the type of communication. Standard deviations are shown in parentheses. $\%\Delta\overline{eComms}$ quantifies the variation of $\overline{eComms}$ from scenarios using LoRa to scenarios using NB-IoT.

deploy			
5 *measurement* ONs			
	LoRa (J)	NB-IoT (J)	$\%\Delta\overline{eComms}$
1min uptime	98,34 (30,95)	115,24 (36,08)	-17.19
2min uptime	98,75 (32,25)	115,16 (37,65)	-16.62
3min uptime	101,02 (33,35)	117,95 (38,99)	-16.76
15 *measurement* ONs			
	LoRa (J)	NB-IoT (J)	$\%\Delta\overline{eComms}$
1min uptime	271,80 (43,50)	374,44 (58,07)	-37.76
2min uptime	320,73 (58,66)	441,04 (77,78)	-37.51
3min uptime	356,79 (65,93)	487,75 (88,15)	-36.71
30 *measurement* ONs			
	LoRa (J)	NB-IoT (J)	$\%\Delta\overline{eComms}$
1min uptime	513,75 (65,13)	815,97 (95,02)	-58.83
2min uptime	665,64 (79,76)	1055,40 (115,45)	-58.55
3min uptime	823,80 (102,39)	1293,10 (149,06)	-56.97

update			
5 *measurement* ONs			
	LoRa (J)	NB-IoT (J)	$\%\Delta\overline{eComms}$
1min uptime	121,43 (47,24)	136,81 (53,31)	-12.67
2min uptime	121,53 (52,29)	137,09 (59,07)	-12.80
3min uptime	118,77 (47,56)	133,45 (53,55)	-12.36
15 *measurement* ONs			
	LoRa (J)	NB-IoT (J)	$\%\Delta\overline{eComms}$
1min uptime	376,87 (87,62)	456,47 (104,75)	-21.12
2min uptime	425,35 (92,83)	512,65 (110,47)	-20.52
3min uptime	447,44 (116,43)	538,26 (136,96)	-20.30
30 *measurement* ONs			
	LoRa (J)	NB-IoT (J)	$\%\Delta\overline{eComms}$
1min uptime	812,87 (141,42)	1030,84 (174,41)	-26.81
2min uptime	1015,03 (163,06)	1284,27 (202,64)	-26.53
3min uptime	1203,50 (177,83)	1511,64 (216,01)	-25.60

As for Section IV-A, combining the utilization of an RN with longer uptime duration increases $\overline{eDynamic}$ in all scenarios. Table II shows that for the smallest cluster size, going from 1 to 3 min while using an RN increases $\overline{eDynamic}$ from 104,24 to 109,81 J for *deploy*, and from 111,13 to 136,31 J for *update*. For the largest cluster size, $\overline{eDynamic}$ increases from 450,39 to 483,82 J for *deploy*, and from 505,62 to 665,37 J for *update*.

Radio technology: Table IV presents $\overline{eComms}$ for *deploy* and *update*, according to the number of *measurement* ONs, the uptime duration and the simulated radio technology (*i.e.*, LoRa, NB-IoT).

For *deploy*, the smallest cluster size and 1 min uptime duration, using NB-IoT instead of LoRa increases $\overline{eComms}$ by 17,19% (from 98,34 to 115,24 J). For the largest cluster size, $\overline{eComms}$ increases by 58,83% (from 513,75 to 815,97 J). For *update*, the smallest cluster size and 1 min uptime duration, using NB-IoT instead of LoRa increases $\overline{eComms}$ by 12,67% (from 121,43 to 136,81 J). For the largest cluster size, $\overline{eComms}$ increases by 26,81% (from 812,87 to 1030,84 J).

Results show that in our use-case NB-IoT has a

higher consumption than LoRa under any scenario. *Deploy/update* duration using LoRa or NB-IoT are not shown in this paper, but NB-IoT's higher bandwidth implies in average a negligible reduction of *deploy/update* duration (less than 3% in few scenarios). This is due to the small size of coordination exchanges between ONs. Thus, for our use-case, LoRa's lower energy consumption is a better choice for communications.

V. State of the art

Works regarding DTNs address challenges related to our use-case, such as systems where nodes have intermittent contacts between each other [8]–[10]. Among these contributions, some of them specifically address the problem of communication between nodes at 1-hop with asynchronous (or random) wake-up schedules [11]. These works usually leverage wake-up schedules of nodes to find a time/energy trade-off for message exchanges. These contributions however do not specifically apply to our use-case, as non-synchronized and non-controllable uptime schedules are considered.

Among contributions dealing with observatory CPS [12]–[15], very few environments impose hard constraints on the CPS such as the AT (*i.e.*, combination of a lack of connectivity with the external world, limited infrastructure and energetic budget). Among contributions dealing with observatory CPS specifically in the AT, none specifically addresses the coordination of changes during adaptation. However, the following contributions are complementary with this paper, as they deal with the dissemination of data (*e.g.*, update data) in CPS deployed in the AT.

In [5], authors conducted experiments for disseminating update data to ONs. The total ONs uptime duration required to complete the dissemination according to different uptime duration is presented. The studied use-case is similar to ours: one sender communicating with multiple receivers. However, ONs are either always-up or have synchronized wake-ups. In [1] authors studied different policies of data dissemination for the DAO-CPS. The use-case topology is also similar: one sender disseminates data to multiple non-synchronized receivers. Policies are: extending the uptime duration of both sender and receiver until the end of data transmission, or/and hinting to receivers of the next uptime of the sender, to facilitate future transmission. While the extended uptime policy is not relevant in our use-case because of the very small size of exchanged messages between ONs, the hint policy can be considered as a future work.

Our contribution aims at combining the extreme conditions of the AT with the coordination of adaptation tasks in a CPS deployed in such an environment. Understanding ONs energy consumption for different scenarios under plausible assumptions is crucial for better anticipation of ONs energy consumption in real deployments.

VI. Conclusion

CPSs deployed in environments like the AT for sensing and observation face extreme conditions. Nodes composing such CPSs are forced to sleep most of the time to save energy and increase their lifetimes. To enable collaboration, services hosted can be coupled between nodes. To prevent failure, the deployment or update of such services has to be safely coordinated. This paper aims at evaluating and studying the impact on energy consumption of nodes during coordination. Two coordination cases (deployment, update) and plausible scenarios (number of nodes, uptime duration, radio technology, relay node availability) are simulated.

Uptime schedules are generated to simulate nodes sleeping behaviors. To get a representative set of results, each scenario is run over 200 uptime schedules. For each scenario, an average of the energy consumption is given and discussed.

Results show that nodes waking-up specifically for coordination implies non-realistic energy consumption when a relay node is not available. Taking advantage of uptimes reserved for sensing or observations only adds the energy consumption overhead induced by the execution of *deploy/update* tasks and by communications. LoRa is the best choice for communications under any scenario due to the very small sizes of exchanged coordination data. Finally, having an available relay node is always favorable with regards to energy consumption.

Future works aim at identifying and studying leverages to optimize the energy consumption of coordination between sleeping ONs. More precise scenarios are considered: simulating weaker RNs with intermittent availability and calibrating future simulations on real *deploy/update* tasks.

Acknowledgment

The DAO project is supported by the Research Council of Norway (RCN) IKTPluss program, project number 270672.

References

[1] I. Rais, L. Guegan, and O. Anshus, "Impact of loosely coupled data dissemination policies for resource challenged environments," in *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. Taormina, Italy: IEEE, May 2022, pp. 524–533. [Online]. Available: https://ieeexplore.ieee.org/document/9826006/

[2] I. Raïs, O. Anshus, J. M. Bjørndalen, D. Balouek-Thomert, and M. Parashar, "Trading data size and CNN confidence score for energy efficient CPS node communications," in *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, pp. 469–478.

[3] L. Guegan, I. Rais, and O. Anshus, "Validation of esds using epidemic-based data dissemination algorithms," in *2023 19th Annual International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, 2023.

[4] M. Chardet, H. Coullon, and S. Robillard, "Toward safe and efficient reconfiguration with concerto," vol. 203. [Online]. Available: https://hal.inria.fr/hal-03103714

[5] R. Tollefsen, I. Rais, J. M. Bjørndalen, P. Hoai Ha, and O. Anshus, "Distribution of Updates to IoT Nodes in a Resource-Challenged Environment," in *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, May 2021, pp. 684–689.

[6] S. Tofaily, I. Rais, and O. Anshus, "Quantifying the variability of power and energy consumption for iot edge nodes," in *2023 19th Annual International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, 2023.

[7] I. Raïs, J. M. Bjørndalen, P. Hoai Ha, K.-A. Jensen, L. S. Michalik, H. Mjøen, Tveito, and O. Anshus, "UAVs as a Leverage to Provide Energy and Network for Cyber-Physical Observation Units on the Arctic Tundra," in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, May 2019, pp. 625–632, iSSN: 2325-2944.

[8] A. Verma, Savita, and S. Kumar, "Routing Protocols in Delay Tolerant Networks: Comparative and Empirical Analysis," *Wireless Personal Communications*, vol. 118, no. 1, pp. 551–574, May 2021. [Online]. Available: https://doi.org/10.1007/s11277-020-08032-4

[9] A. Keshavarzian, H. Lee, and L. Venkatraman, "Wakeup scheduling in wireless sensor networks," in *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, ser. MobiHoc '06. New York, NY, USA: Association for Computing Machinery, May 2006, pp. 322–333. [Online]. Available: https://doi.org/10.1145/1132905.1132941

[10] R. Su, R. Venkatesan, and C. Li, "A new node coordination scheme for data gathering in underwater acoustic sensor networks using autonomous underwater vehicle," in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2013, pp. 4370–4374, iSSN: 1558-2612.

[11] R. Zheng, J. C. Hou, and L. Sha, "Asynchronous wakeup for ad hoc networks," in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*. Annapolis Maryland USA: ACM, Jun. 2003, pp. 35–45. [Online]. Available: https://dl.acm.org/doi/10.1145/778415.778420

[12] S. e. a. Guo, "The application of the internet of things to animal ecology," *Integrative zoology*, 2015.

[13] N.-S. Kim, K. Lee, and J.-H. Ryu, "Study on iot based wild vegetation community ecological monitoring system," in *Seventh International Conference on Ubiquitous and Future Networks*, 2015.

[14] S. Lin, F. Lyu, and H. Nie, "An automatic instrument integration scheme for interoperable ocean observatories," *Sensors*, 2020.

[15] I. Talzi, A. Hasler, S. Gruber, and C. Tschudin, "Permasense: investigating permafrost with a wsn in the swiss alps," in *Proceedings of the 4th workshop on Embedded networked sensors*, 2007.