SEASON - Towards Safe Reconfiguration for Sound Observation Network

Jolan Philippe
Université de Rennes, IRISA, UMR 6074
F-35000 Rennes, France
Email: jolan.philippe@rennes.fr

Antoine Bernabeu
Polytech, LS2N, UMR 6004
F-44000 Nantes, France
Email: antoine.bernabeu@univ-nantes.fr

***Abstract*—Wildlife monitoring sensors must be deployed in remote, often inaccessible environments for extended periods without human intervention. To remain autonomous and relevant, these embedded systems must be reconfigurable (*e.g.*, adapting to seasonal changes, energy constraints, or evolving ecological missions). In this paper, we present Season (Safe rEconfigurAtion for Sound Observation Network), a vision for safe, remote, and coordinated reconfiguration of eco-acoustic monitoring devices powered by intermittent energy. Season introduces a reconfigurable kernel capable of applying both full and partial application updates, driven by a model-based coordination engine. Our approach builds upon the Concerto formalism to provide verifiable guarantees about the execution of distributed reconfiguration plans, even in the presence of critical timing and energy constraints. We discuss our prototype implementation and its deployment on energy-harvesting devices, and reflect on the challenges of scaling to a fleet of CPS with formal reasoning under model-checking constraints.**

***Index Terms*—Intermittent device, Reconfiguration, Safety, Cyber-Physical System**

I. Introduction

Eco-acoustic monitoring systems are emerging as a promising solution for studying and preserving biodiversity in remote environments. These systems leverage *Internet of Things (IoT)* technologies to collect acoustic data over long periods, enabling ecological research without requiring constant human intervention. Instead of relying on manual sensor redeployment or maintenance, eco-acoustic devices are designed to operate autonomously, sometimes in extreme or isolated conditions such as arctic zones [1] or dense forests [2]. This paradigm supports non-intrusive data collection, preserving the natural behavior of wildlife while ensuring continuous monitoring. To reduce energy usage, such devices may dynamically adjust their sensing parameters (e.g., sampling rate, recording frequency) or even switch missions depending on the season or ecological focus.

Recent developments in *intermittent computing* and the use of *supercapacitors* enable these devices to operate without batteries [3], tolerating power failures by saving their state between energy harvesting cycles. Nevertheless, managing and adapting these constrained, decentralized systems poses several challenges, particularly when configurations must change over time (e.g., detecting different species or adjusting acoustic resolution). These systems must be considered *critical*: a failure in reconfiguration or operation can lead to significant loss of ecological data, with consequences that might remain undetected for extended periods due to intermittent activity. This calls for *remote, coordinated, and energy-aware reconfiguration*, where embedded intelligence within the network drives autonomous and efficient behavior adaptation without human presence.

In this paper, we present our Season (Safe rEconfigurAtion for Sound Observation Network) vision towards a generic kernel for safe reconfiguration of sensor networks under intermittent power. To this end, we aim at addressing the following challenges:

1) Designing a *reconfigurable kernel* capable of changing the application logic, such as updating sensing or a treatment process over time.
2) Supporting the *safe and remote orchestration of reconfiguration* for distributed and intermittent devices. This includes the ability to schedule and coordinate updates without compromising data continuity or system integrity.
3) Enabling *formal reasoning* over the reconfiguration process, which requires a well-defined model of both the kernel (e.g., inspired by platforms like Trampoline [4] or systems like Contiki-OS [5]) and the transitions involved in the reconfiguration itself. Such a model must support verifiable guarantees such as termination, correctness, and safety of the operation

For validation purposes, we have designed and implemented a concrete implementation of the Season kernel that can be used for an autonomous acoustic device. For evaluation, we plan to evaluate the relevance and the safety of the Season solution on both simulated IoT devices with EnOSlib [6] on top of Chameleon-Edge [7], and on a physical device using microcontrollers such as MSP430FR [8] series from Texas Instrument which have been widely used in the intermittent research community.

II. Motivation and Background

A. Motivation and approach

Embedded systems or *cyber-physical systems (CPS)* must support *reconfiguration* throughout their lifespan to adapt to evolving missions, environmental conditions, or operational constraints. However, due to their deployment in *remote* or *sensitive environments*, such as eco-acoustic sensors in dense forests or arctic zones [1], [2], direct human access is often infeasible or undesirable. Minimizing human presence also reduces disturbance to wildlife, helping to preserve their natural behavior and habitats. These systems are typically *energy-constrained*, mostly powered by batteries, sometimes solar-charged to extend their lifespan with energy availability varying due to weather and environmental conditions. As a result, operational planning must be adapted to consider *limited and variable energy*.

To remain operational over long deployments, these systems must be *fully autonomous*, supporting remote installation, configuration, and update. In this work, we focus on autonomous acoustic devices that monitor animal activity. Their reconfiguration may involve changing the sensing mission (*e.g.*, switching to detect other animal species) or adapting local acquisition parameters (*e.g.*, lowering audio resolution or sample rate [9]) to reduce energy consumption. More generally, reconfigurations can impact the availability of components (e.g., sensor, storage), requiring *coordinated updates* to preserve data flow and system consistency.

These are *critical systems*: a failure or misconfiguration could interrupt data collection, with consequences that may remain undetected for extended periods due to intermittent operation. Ensuring the *safety* of reconfiguration, through verification of its correctness, termination, and energy feasibility, is therefore essential before deployment.

B. Autonomous acoustic device

The science of eco-acoustics aims to study the role of sound in ecological processes such as population dynamics, species assemblages, ecosystem decline and restoration, and the emergence of a sonic landscape or ”soundscape” [10]. It was only in recent years that eco-acoustics emerged as a powerful quantitative method for surveying biodiversity at the planetary scale [11].

Passive acoustic monitoring (PAM) of fauna has been widely developed in recent years [12]. Contrary to traditional point counts carried out by field operators, they allow the simultaneous monitoring of numerous stations, at any time of the day, over long periods, in poorly accessible locations, with minimal disturbance in the field. Sound storage also enables numerous subsequent laboratory analyses. Today, there are several sensors with different prices, battery life, and storage capacities, among which the most popular in field ecology are the Wildlife Acoustic devices Song Meter (SM), AudioMoth, and Swift Recorder [13]. Despite recent technical improvements and decreases in prices, as well as the possibility to schedule records over defined periods, these sensors are still battery-intensive and require large storage capacities. In addition to ecological problems (regular replacement of batteries), long-term acoustic monitoring is thus constrained by the volume of storage required for the acoustic data collected, but also the need to return regularly to the field to renew batteries and retrieve the stored data. This last point greatly diminishes the interest of PAM, one of whose strengths is precisely not to generate any disturbance in the field.

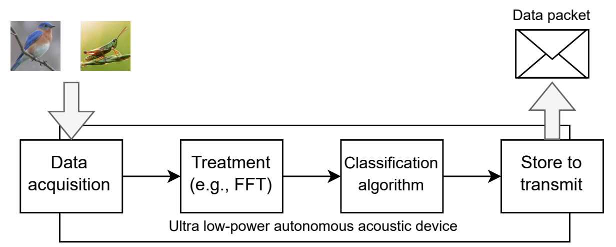


Fig. 1: Architecture of a sensor node for wildlife monitoring, depending on the mission, the node can target either insects or birds.

To overcome the inconvenience associated with PAM, recent research is advocating the development of fully autonomous, or even intermittent [3], sensors. The architectural design of such a node is illustrated in Figure 1. Data is collected and processed on the device, while results are transmitted using a wireless medium. Processing data at the edge will significantly reduce the amount of data that needs to be sent from the device to the gateway, and even reduce the size of the databases needed.

Looking ahead, the benefits of being able to reconfigure these systems are clear. A fully autonomous system should be capable of reconfiguring itself for different missions throughout its lifespan, thereby reducing the need for human intervention even further. Continuing with the eco-acoustic example, depending on the season, different bird species can be monitored. This means that the system can be reconfigured to adapt the application, modifying, for example, the detection or classification algorithm.

Previous work has mainly focused on intermittent devices with wireless communication capabilities. For example, firmware updates have been tested on intermittent devices using Bluetooth [14]. On the other hand, a framework for over-the-air updates on energy-harvesting-powered systems has been proposed [15], [16]. These works demonstrate that communication with an intermittent system is possible and that it is therefore feasible to update all or part of the onboard firmware. However, this work focuses solely on the communication medium and does not include orchestration of the reconfiguration.

C. Safe reconfiguration

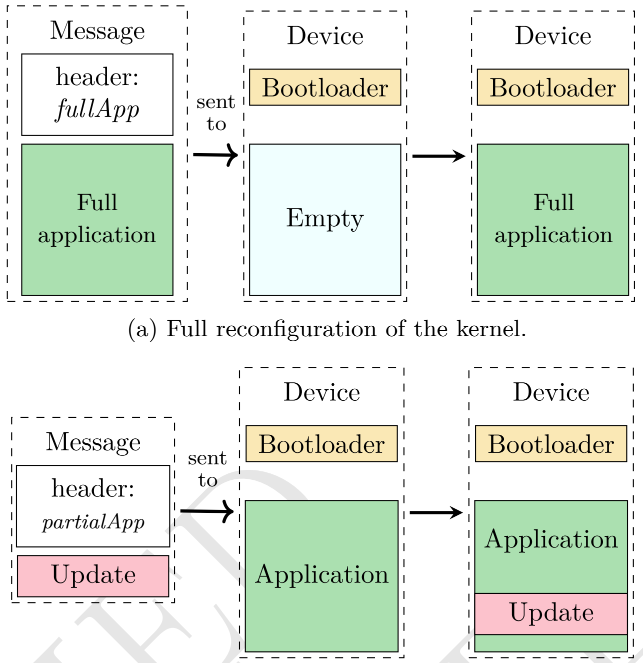
Concerto [17] is a reconfiguration model designed to manage and reconfigure system components through their lifecycle. Each Concerto component acts as a proxy around a piece of software developed as an alternative to control scripts for tasks such as installation, maintenance, and service suspension. The topological interface of components is specified through *provide* ports and *use* ports: *provide* ports that indicate services or data offered by the component when active, while *use* ports specify the dependencies or requirements needed by the component during active phases. Internally, components consist of places, representing lifecycle milestones, and transitions, corresponding to concrete actions (e.g., a piece of code) to be executed. Ports are linked to specific subsets of these lifecycle elements called groups, denoting the lifecycle stages during which ports remain active. Additionally, components are defined by behaviors, which are subsets of transitions. At runtime, each component instance maintains a queue of behaviors, executing requested behaviors sequentially. Similar to, yet distinct from Petri nets, tokens traverse through places and transitions according to the active behavior, activating and deactivating ports as they enter or exit corresponding groups. A component cannot reach a place linked to a *use* port connected with an inactive *provide* port; Similarly, a component cannot leave a place linked to a *provide* port connected to an active *use* port.

Concerto's formal model [17], [18] enables reasoning about reconfiguration processes, ensuring through model-checking techniques the termination of reconfiguration operations derived from a specific plan. Besides reasoning opportunities, Concerto performance in real-world applications [19], [20] is efficient as demonstrated in practical scenarios. Recent versions of Concerto support intermittent nodes by integrating message brokers to manage communication with sleeping nodes, making it particularly suitable for batteryless components [21]. Figure 3 presents illustrative examples of gateway and sensor lifecycles: the gateway comprises 6 places, 8 transitions, and 3 ports (2 *provide* ports, 1 *use* port), while the sensor contains 7 places, 9 transitions, and 3 ports (1 *provide* port, 2 *use* ports). To interact with the components, Concerto includes a language with a few possible instructions, such as `pushB` and `wait`. The `pushB` instruction is non-blocking, enqueuing behaviors for sequential execution, whereas `wait` is a blocking command, ensuring a specific behavior completes before progressing further. An example of usage of these instructions can be seen in Listings 1 and 2, illustrating reconfiguration programs for system deployment and sensor mission updates, respectively.

III. The SEASON approach

A. Reconfigurable kernel

Within the Season approach, we plan to have a reconfigurable kernel to support two complementary reconfiguration strategies, as illustrated in Figure 2. First, *full reconfiguration* consists of transmitting an entire application to an initially blank device, where only the bootloader is present. This approach is typically used during the initial setup phase or when a complete application replacement is needed. In contrast, a *partial update* enables more fine-grained modifications by sending only the segments of code that need to be replaced. The update message includes metadata (e.g., memory ranges) to identify which parts should be modified. Introducing such an incremental mechanism aims at minimizing energy usage related to data transfer.



(a) Full reconfiguration of the kernel.

(b) Update reconfiguration of the kernel.

Fig. 2: Perspective of a reconfigurable device with a message passing approach.

B. Driving safe reconfiguration

a) Gateway lifecycle behaviors: The gateway lifecycle illustrated in Figure 3 is defined from Concerto behaviors as described as follows:

- `setup`: Activates the communication channel using a wireless medium.
- `deploy`: Transmits either a complete application (left path) or a subpart (right path). Once the sensor fully receives the application, the gateway begins listening to incoming sensor data.
- `update`: Temporarily stops data listening to send a partial update, including code fragments and replacement indexes, to the sensor.
- `stop`: Ceases listening to facilitate transmission of a complete application replacement.
- `shutdown`: Deactivates the communication channel.

Listing 1: Example of reconfiguration plan in CONCERTO for deploying a sensor along with a gateway.

```
pushB(gateway, setup)
pushB(sensor, setup)
wait(sensor, setup)
pushB(gateway, deploy)
pushB(sensor, deploy)
```

Listing 2: Example of reconfiguration plan in CONCERTO for updating a sensor connected to a gateway.

```
pushB(gateway, update)
pushB(sensor, interrupt)
wait(sensor, interrupt)
pushB(gateway, deploy)
pushB(sensor, deploy)
```

b) Sensor lifecycle behaviors: The sensor lifecycle illustrated in Figure 3 is defined from Concerto behaviors as described as follows:

- `setup`: First, it runs its bootloader, then activates the communication channel using a wireless protocol.
- `deploy`: Receives either a complete application (left path) or a subpart (right path). Once the *sensor* fully receives the application, it transits to an intermediate state before being operational (*i.e.*, *running*).
- `interrupt`: Stops the *sensor* service, it is not listening nor sending data to the *gateway*.
- `wipe`: Erases its full application content, and waits for a new application to receive.
- `destroy`: The device is put in a deep sleep mode waiting for an external input (from wireless communication) to wake up.

c) Executing a safe reconfiguration: We consider two scenarios to illustrate safe reconfigurations driven by CONCERTO. Listing 1 gives the reconfiguration program to deploy a *sensor* to capture sound information, alongside a *gateway* in charge of sending the application to the sensor, and collecting data. The second scenario, illustrated in Listing 2, consists of changing the *sensor*'s mission. This update is triggered from the *gateway* side. Note that the programs have been written by hand, but could be synthesized using SMT solvers [22], ensuring the correctness of programs from specified goals.

In the following, we describe the state of each configuration, in the first scenario (*i.e.*,, deployment), according to its number and decompose each state. Figure 3 gives a visual representation of the reached states.

0. Both the *gateway* and the *sensor* are in their initial positions. The `setup` behavior is added to the queue of the two components;
1. The *gateway* starts its transition linked to the `setup` behavior;
2. The *gateway* ends its previously triggered transition and is now ready to send a full application. It enables its related provide port `appFull`. Since the behavior `setup` is fully executed, it can be removed from the *gateway*'s queue;
3. The *sensor* starts its transition linked to the `setup` behavior;
4. The *sensor* ends the transition and is now ready to receive a full application. It starts using the *gateway*'s provide port with its use port `gwFull`;
5. The behavior `deploy` is added to both queues and starts to be executed. Concretely, the *gateway* is sending the app while the *sensor* receives it;
6. The app is fully sent by the *gateway* and then received by the *sensor*;
7. The deployment continues by charging the app in the *sensor* memory. The *sensor* is not using the *gateway*'s provide port anymore;
8. The *sensor* finishes its deployment by running the application;
9. Since the *sensor* is now fully deployed, it activates its provide port *service* to allow the *gateway* to continue its deployment;
10. The *gateway* triggers its last transition to start its listening channel to get data collected by the sensor;
11. The *gateway* is fully deployed, and uses the *sensor*'s provide port using a use port.

IV. DISCUSSION

We previously described the core of our vision of SEASON for reconfiguring autonomic sensors. However, there are still some interesting medium-term challenges, notably when integrating our solution into open source ecosystem(s).

A. Prototype Implementation

As a first step toward validating the SEASON approach, we have developed an initial prototype of our reconfigurable kernel. This prototype targets a low-power acoustic device operating under intermittent power conditions. The system includes a bootloader and a minimal runtime that supports both full deployments and incremental updates, as described in Section III-A. The prototype currently enables remote application deployment, manages transition states using CONCERTO's coordination model, and supports message-driven reconfiguration.

Our current implementation supports receiving application fragments, storing them in flash memory, and activating them through controlled state transitions. This early implementation validates the feasibility of combining intermittent computing with CONCERTO-based reconfiguration. It serves as a foundation for integrating further safety and verification mechanisms and testing SEASON 's scalability across multiple distributed devices.

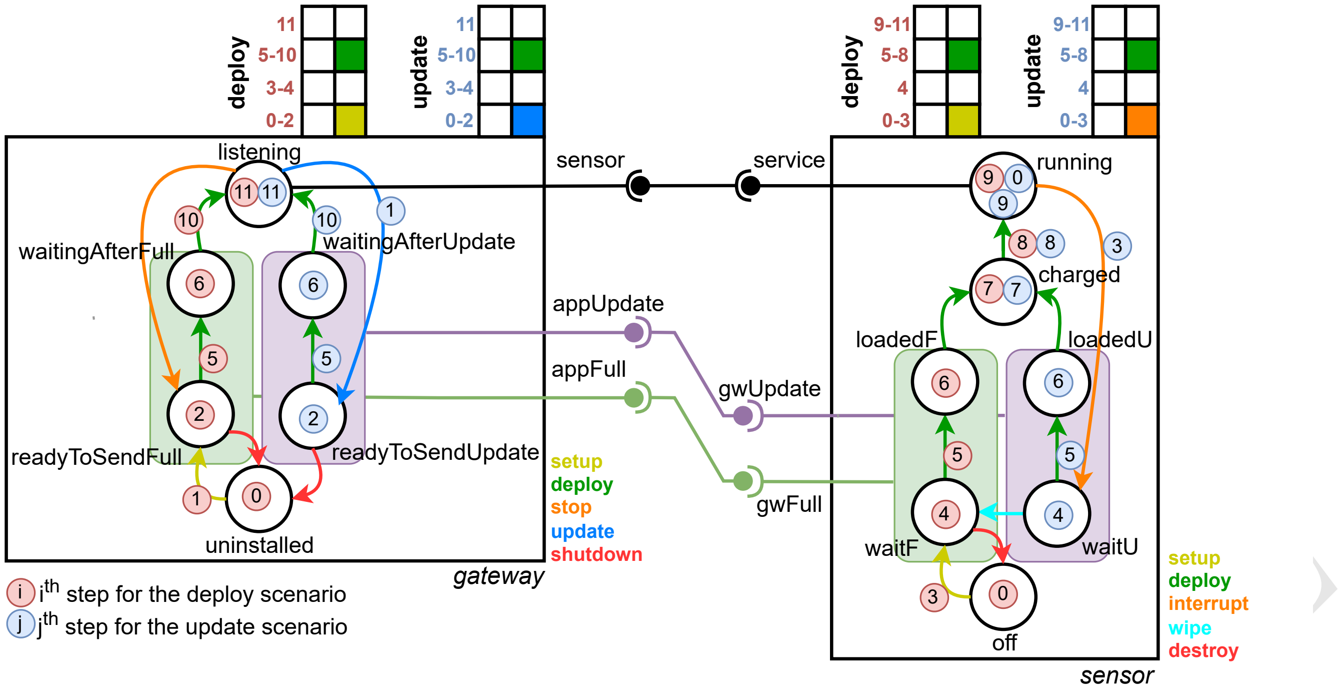


Fig. 3: Control components in Concerto of a gateway and a sensor. States during the deployment (resp. update) operation are represented in red (resp. blue) when applying the reconfiguration plan of Listing 1 (resp. Listing 2).

The source code corresponding to components on CONCERTO and the core to be loaded on the sensor is available online [1].

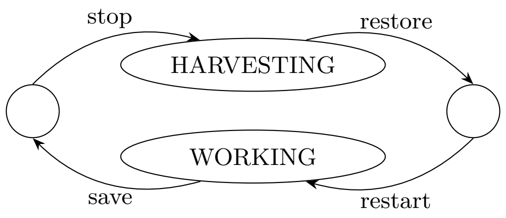
B. Coordination of a fleet of CPS

The SEASON vision targets more than the reconfiguration of individual nodes to support the coordination of entire fleets of CPS, each potentially executing distinct missions. In this context, each sensor is paired with a gateway (*e.g.*, implemented as a dedicated thread or lightweight process), and all gateways interface with a central backend infrastructure. This infrastructure would be responsible for aggregating, processing, and storing collected data, typically involving shared resources such as databases. Dependencies across numerous components must be managed carefully to ensure the absence of inconsistent states. To support such a property, SEASON will take advantage of Component-Based Software Engineering (CBSE) technologies and their established reconfiguration models [23]. These models offer modular reasoning and interface contracts—features that can be leveraged to ensure safe reconfiguration in CPS networks. This enables the use of model-checking techniques to validate the behavior of coordinated systems, as demonstrated in earlier work [24], [25]. However, increasing the number of CPS also increases the size of the global reconfiguration model. This scalability concern is particularly challenging when applying formal verification techniques, such as model-checking. In particular, approaches based on Linear Temporal Logic (LTL) can become computationally expensive at scale. Addressing this challenge requires adapting model-checking methods to cope with the state space explosion.

[1]https://anonymous.4open.science/r/ICSME25-NIER-SEASON

C. Modeling energy-aware lifecycle

Beyond functional aspects, the SEASON approach aims at incorporating non-functional properties, with a focus on energy-awareness for intermittent devices. In addition to the functional lifecycle presented in Section III-B, we propose introducing a *transversal lifecycle* that captures the energy state of the system. This lifecycle will distinguish two primary states: *working*, when the device is powered and operational, and *harvesting*, when energy is insufficient to continue execution.



To enable energy-aware decision-making, we plan to design an energy model that estimates the energy cost associated with each reconfiguration action. When a given action exceeds the device's current energy resource, the model will decide whether to perform a state save and transition to the harvesting state. Once sufficient energy has been harvested, the device can resume execution by restoring the previously saved state.

References

[1] L. Guegan and I. Rais, " Simulation of Distributed Systems in Constrained Environments Using ESDS: the Arctic Tundra Case ," in *2023 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing Communications (GreenCom) and IEEE Cyber, Physical Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*. Los Alamitos, CA, USA: IEEE Computer Society, Dec. 2023, pp. 547–554. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics60724.2023.00104

[2] X. Liu, T. Yang, and B. Yan, "Internet of things for wildlife monitoring," in *2015 IEEE/CIC International Conference on Communications in China - Workshops (CIC/ICCC)*, 2015, pp. 62–66. [Online]. Available: https://doi.org/10.1109/ICCChinaW.2015.7961581

[3] V. Lostanlen, A. Bernabeu, J.-L. Béchennec, M. Briday, S. Faucou, and M. Lagrange, "Energy efficiency is not enough:towards a batteryless internet of sounds," in *Proceedings of the 16th International Audio Mostly Conference*, 2021. [Online]. Available: https://doi.org/10.1145/3478384.3478408

[4] J.-L. Béchennec, M. Briday, S. Faucou, and Y. Trinquet, "Trampoline - an open source implementation of the osek/vdx rtos specification," in *11th Int. Conf. on Emerging Technologies and Factory Automation (ETFA'06)*. Prague, Czech Republic: IEEE, Sep. 2006. [Online]. Available: https://inria.hal.science/inria-00538492

[5] C. Durmaz, M. Challenger, O. Dagdeviren, and G. Kardas, "Modelling contiki-based iot systems," in *6th symposium on languages, applications and technologies (SLATE 2017)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017, pp. 5–1. [Online]. Available: https://drops.dagstuhl.de/entities/document/10.4230/OASIcs.SLATE.2017.5

[6] B. Jonglez, M. Simonin, J. Philippe, and S. M. Kaddour, "Multi-provider capabilities in enoslib: driving distributed system experiments on the edge-to-cloud continuum," in *Distributed Applications and Interoperable Systems (DAIS 2025), Held as Part of the 20th International Federated Conference on Distributed Computing Techniques, DisCoTec 2025, Lille, France, June 16–20, 2025, Proceedings*. Lille, France: Springer LNCS-IFIP, jun 2024. [Online]. Available: https://inria.hal.science/hal-05052776v1/file/enoslib_dais_2025_v3.pdf

[7] K. Keahey, J. Anderson, M. Sherman, Z. Zhen, M. Powers, I. Brunkan, and A. Cooper, "Chameleon@ Edge Community Workshop Report," 2021.

[8] "MSP430FR5994 datasheet | TI.com." [Online]. Available: https://www.ti.com/document-viewer/msp430fr5994/datasheet

[9] A. Benoit, L. Lefèvre, A.-C. Orgerie, and I. Raïs, "Reducing the energy consumption of large-scale computing systems through combined shutdown policies with multiple constraints," *The International Journal of High Performance Computing Applications*, vol. 32, no. 1, pp. 176–188, 2018. [Online]. Available: https://doi.org/10.1177/1094342017714530

[10] J. Sueur and A. Farina, "Ecoacoustics: the Ecological Investigation and Interpretation of Environmental Sound," *Biosemiotics*, vol. 8, no. 3, pp. 493–502, Dec. 2015. [Online]. Available: https://doi.org/10.1007/s12304-015-9248-x

[11] D. Stowell and J. Sueur, "Ecoacoustics: acoustic sensing for biodiversity monitoring at scale," *Remote Sensing in Ecology and Conservation*, vol. 6, Aug. 2020. [Online]. Available: https://doi.org/10.1002/rse2.174

[12] L. S. M. Sugai, T. S. F. Silva, J. W. Ribeiro, and D. Llusia, "Terrestrial Passive Acoustic Monitoring: Review and Perspectives," *BioScience*, vol. 69, no. 1, pp. 15–25, Jan. 2019. [Online]. Available: https://academic.oup.com/bioscience/article/69/1/15/5193506

[13] M. Toenies and L. Rich, "Advancing bird survey efforts through novel recorder technology and automated species identification," *California Fish and Wildlife Journal*, vol. 107, no. 2, pp. 56–70, Aug. 2021. [Online]. Available: https://nrm.dfg.ca.gov/FileHandler.ashx?DocumentID=193712&inline

[14] J. de Winkel, H. Tang, and P. Pawełczak, "Intermittently-powered bluetooth that works," in *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*, ser. MobiSys '22. New York, NY, USA: Association for Computing Machinery, Jun. 2022, pp. 287–301. [Online]. Available: https://doi.org/10.1145/3498361.3538934

[15] W. Wei, C. Pan, S. Islam, J. Banerjee, S. Palanisamy, and M. Xie, "Intermittent OTA Code Update Framework for Tiny Energy Harvesting Devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10569023/

[16] W. Wei, J. Banerjee, S. Islam, C. Pan, and M. Xie, "Energy-aware Incremental OTA Update for Flash-based Batteryless IoT Devices," Jun. 2024, arXiv:2406.12189 [cs]. [Online]. Available: http://arxiv.org/abs/2406.12189

[17] M. Chardet, H. Coullon, and S. Robillard, "Toward Safe and Efficient Reconfiguration with Concerto," *Science of Computer Programming*, 2021. [Online]. Available: https://doi.org/10.1016/j.scico.2020.102582

[18] F. Arfi, H. Coullon, F. Loulergue, J. Philippe, and S. Robillard, "An Overview of the Decentralized Reconfiguration Language Concerto-D through its Maude Formalization," in *Electronic Proceedings in Theoretical Computer Science*, ser. Electronic Proceedings in Theoretical Computer Science, Groningen, Netherlands, Jun. 2024, pp. 1–18, satellite workshop of DisCoTec 2024. [Online]. Available: https://inria.hal.science/hal-04572043

[19] M. Chardet, H. Coullon, and C. Pérez, "Predictable Efficiency for Reconfiguration of Service-Oriented Systems with Concerto," in *CCGrid 2020 : 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing*, 2020. [Online]. Available: https://doi.org/10.1109/CCGrid49817.2020.00-59

[20] J. Philippe, A. Omond, H. Coullon, C. Prud'Homme, and I. Raïs, "Fast Choreography of Cross-DevOps Reconfiguration with Ballet: A Multi-Site OpenStack Case Study," in *SANER 2024 - IEEE International Conference on Software Analysis, Evolution and Reengineering*. Rovaniemi, Finland: IEEE, Mar. 2024, pp. 1–11. [Online]. Available: https://doi.org/10.1109/SANER60148.2024.00007

[21] A. Omond, H. Coullon, I. Raïs, and O. Anshus, "Leveraging Relay Nodes to Deploy and Update Services in a CPS with Sleeping Nodes," in *Proceeding of the 2023 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*. Danzhou, China: IEEE, Dec. 2023, pp. 1–8. [Online]. Available: https://hal.science/hal-04372320

[22] S. Robillard and H. Coullon, "SMT-Based Planning Synthesis for Distributed System Reconfigurations," in *FASE 2022 : 25th International Conference on Fundamental Approaches to Software Engineering*, 2022. [Online]. Available: https://doi.org/10.1007/978-3-030-99429-7_15

[23] H. Coullon, L. Henrio, F. Loulergue, and S. Robillard, "Component-Based Distributed Software Reconfiguration: a Verification-Oriented Survey," *ACM Computing Surveys*, vol. 56, no. 1, pp. 1–37, Jan. 2024. [Online]. Available: https://inria.hal.science/hal-04067909

[24] D. Lime and O. H. Roux, "Model checking of time petri nets using the state class timed automaton," *Discrete Event Dynamic Systems*, vol. 16, no. 2, p. 179–205, Apr. 2006. [Online]. Available: https://doi.org/10.1007/s10626-006-8133-9

[25] E. André, J.-L. Béchennec, S. Chattopadhyay, S. Faucou, D. Lime, D. Marinho, O. H. Roux, and J. Sun, "Verifying timed properties of programs in iot nodes using parametric time petri nets," in *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing*, ser. SAC '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 1998–2006. [Online]. Available: https://doi.org/10.1145/3672608.3707861