Delft University of Technology
Master's Thesis in Embedded Systems

Wake-Up Alignment for Batteryless Sensors

Carlo Delle Donne

TU Delft Delft University of Technology

Wake-Up Alignment for Batteryless Sensors

Master's Thesis in Embedded Systems

Embedded Software Section
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands

Carlo Delle Donne
c.delledonne@student.tudelft.nl
carlo.delle.donne@gmail.com

12th November 2018

Author
Carlo Delle Donne

Title
Wake-Up Alignment for Batteryless Sensors

MSc presentation
13th November 2018

Graduation Committee

Prof. Dr. Koen G. Langendoen (chair)	Delft University of Technology
Dr. Przemysław Pawełczak (supervisor)	Delft University of Technology
Dr. Sebastian T. Erdweg	Delft University of Technology

Abstract

Batteryless network nodes could enable unprecedented ubiquitous sensing applications, while gathering energy from the environment and hence removing maintenance costs for batteries. However, these devices inherently suffer frequent power failures and reboots, due to the volatility of the incoming energy, thus classical synchronous message passing cannot work. On the other hand, the asynchronous scheme is not viable either, since power outages can occur at different rates on two devices, and for unpredictable lengths of time. In order to align wake-ups of wireless nodes, a non-canonical mechanism to keep track of time must be devised. We present a novel architecture for reliable packet exchange between energy-harvesting devices, resilient to power failures and agnostic about the physical layer. The proposed architecture, implemented on real hardware, shows an improvement in throughput of a factor of 10, and reduces energy waste by 33 % to 78 %, as compared to the asynchronous message passing.

Preface

This MSc thesis has been carried out at the TU Delft, within the Embedded Software group, under the supervision of dr. Przemysław Pawełczak. The field of intermittent computing is of particular interest to this group, as it is one very hot topic among the embedded systems community. Nevertheless, wireless communication for energy-harvesting, intermittently-powered sensor nodes had never been investigated so in depth, neither in Delft nor anywhere else. The relevance of such a problem within the context of a sustainable, ubiquitous Internet of things was the main source of inspiration for me to research on this topic. The results obtained are a first stepping stone to transiently-powered wireless sensor networks, and I am glad I could contribute to it.

This project also marks the end of my MSc, as well as the beginning of a new journey at the TU Delft. The technical and personal help of some people, whether they were aware or not, was essential to the development of this work, and to my growth as a student and as a human. All those involved in this project had a fundamental role. I want to thank Amjad Majid, for the technical support, the humble feedback and the friendly collaboration. Sinan Yıldırım, for the sharp criticism and the great help with the literature review. Josiah Hester, for the great opportunity at Northwestern University and the high-quality overseas contribution. Przemek Pawełczak, for the constant feedback, the close supervision and, most of all, for the genuine mentorship concerning all aspects of life. Dimitris Patoukas, for the daily support that went far beyond the university boundaries. Finally, I want to thank the other two members of the graduation committee, prof. Koen Langendoen and dr. Sebastian Erdweg, for accepting to review this dissertation and attend my defense. Outside the academic environment, there are quite a few people I will never be grateful enough to.

Carlo Delle Donne

Delft, The Netherlands
12th November 2018

Contents

Chapter 1

Introduction

The future Internet of things is expected to be populated by billions, even trillions of embedded devices with connectivity functionalities. As of now, most of such devices are powered by batteries, which are expensive, hazardous, bulky, require regular maintenance, and are prone to failure [31]. As the number of embedded nodes deployed around the world increases, relying on battery power for all of them is not a viable solution. Luckily, advances in ultra-low-power microcontrollers and energy harvesters has enabled the possibility to leave batteries behind and to power the swarm of sensors out of ambient energy. Relying on the volatile harvested energy, however, makes computation, communication, and actuation challenging. At the very bottom of this challenge is the *intermittent operation* these low-energy nodes are affected by: the device is only operational as long as the buffered energy is enough to power it, and they experience a power failure as soon as the voltage on the energy buffer drops below some threshold (Figure 1.1). Thus, sensors operate intermittently and they repeatedly lose (i) volatile data and (ii) notion of time.

Researchers have addressed the *intermittent computation* challenge [24] by investigating the problems of preserving forward progress and maintaining data consistency despite power failures. So far, the most researched question is *how frequently volatile data has to be checkpointed to non-volatile memory*, exploring the trade-off between re-execution penalty (reduced by more frequent checkpoints) and checkpoint cost (increasing with the frequency of checkpoints). Hardware-based solutions to intermittent computation problem, such as [17], are less common since they require (obviously) a dedicated hardware, and are not applicable to the current off-the-shelf microcontrollers. Software-based solutions for intermittent computation fall generally under two categories: checkpoint-based and task-based approaches. The former do not require the programmer to perform any transformation on the original source code [3, 2, 35, 25], but they suffer from scalability and efficiency issues. Task-based programming models [8, 27] partly solve these

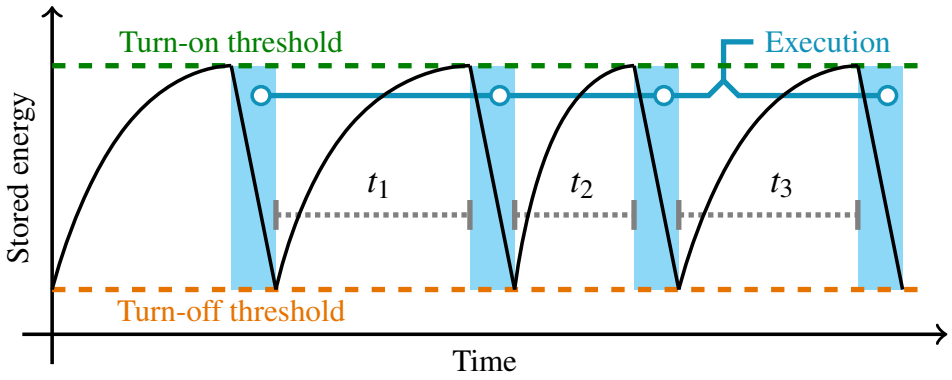


Figure 1.1: Harvested energy induces intermittent execution, whose on and off times are not controllable.

inefficiencies, but they require non-trivial, manual (or compiler-aided) code transformation.

Despite the amount of progress that has been achieved in the intermittent computing domain, reliable *intermittent communication* remains untouched. Intuitively, enabling efficient communication for batteryless, intermittently-powered embedded nodes requires to synchronize their wake-up times. In other words, accurate timestamping and on/off time control mechanisms are needed. Keeping time across power failures is challenging for obvious reasons, while controlling on and off times is bounded by energy availability.

1.1 Challenges of Batteryless Communication

Making any two wireless transceivers communicate with one another, when both are powered by ambient energy and have no access to batteries, is challenging to say the least. As the energy reservoir is very small, it depletes fast after a random power charge from the ambient source, forcing the device to a total shutdown—including any internal clock that must be also powered. Each device then wakes up at a random point in time, and a communication can be established only if both transmitter and receiver are active at the same time. This intermittent (but *synchronized*) communication for batteryless devices has, to the best of our knowledge, not been investigated yet. Beyond the challenge of no control over incoming energy patterns, reliable intermittent communication faces two other core problems described below.

1.1.1 Inadequate or No Timekeeping

Without timekeeping, synchronization becomes very difficult. Embedded devices have on-board timers that reset to zero once power is removed. Thus, standard synchronization approaches would simply not work. Solu-

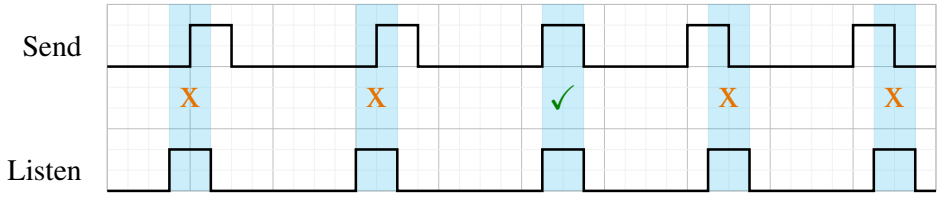


Figure 1.2: Wake-up misalignment. Packets are successfully delivered only when transmitter and receiver are active during the same interval.

tions like real-time clocks are expensive (sometimes cost more than the microcontroller) and require significant board space. Moreover, they do not provide sub-second timekeeping with marginal energy consumption. Capacitive timekeepers [15] are an interesting low-power solution, but they lack accuracy, precision and flexibility, as described in Section 2.2.2.

1.1.2 Network Wake-Up Misalignment

Another challenge is aligning wake-ups across multiple devices in a network. Each device will harvest different amounts of energy, sometimes varying by an order of magnitude, sometimes varying little if at all. This is related to the deployment environment, the orientation, the efficiency of the harvester, and also the behavior of the sensor [14]. This energy availability differential between any two nodes in a network leads to transmission/reception misalignment (Figure 1.2). The kind of wake-up alignment required for the scenario under study differs from the classical synchronization problem, since (i) most of the low-power timekeeping solutions do not provide very good clock accuracy, and (ii) on and off periods are bounded by harvested energy patterns.

1.2 Contributions

To address these critical challenges we provide three main contributions.

1.2.1 Packet Loss Characterization

To begin with, we experimentally analyze the packet loss of two intermittently-powered devices, a transmitter and a receiver, exchanging packets without synchronization. They greedily initiate their radio activity (transmission or reception) as soon as they wake-up after a power failure. It will be shown that, even in most favorable energy arrival conditions, around 90 % of the sent frames are lost due to wake-up misalignment. This preliminary study will demonstrate the need for the next two contributions.

1.2.2 Batteryless Timekeeping Architecture

To address the challenge presented in Section 1.1.1, we propose a new timekeeping architecture, denoted as *hierarchical remanence timekeeper* (HRT), and an accompanying protocol to capture local time on intermittently-powered devices. The core idea is based on remanence (capacitive) timekeepers [15, 16], but some new key features are introduced to improve accuracy, precision and flexibility. In particular, a one-time circuit calibration to ameliorate accuracy and precision, and multiple capacitors combined in a hierarchical array to improve flexibility. Both hardware and software are implemented and tested, showing that these timekeepers can achieve millisecond (even sub-millisecond) resolution.

1.2.3 Intermittent Synchronization Protocol

Complementary to the HRT we propose and experimentally evaluate a protocol for wake-up alignment of two intermittently-powered nodes to enable low-packet-loss message passing. Our method can synchronize transmitter and receiver efficiently by using timing information from the HRT and by applying error correction and sync recovery, to cope with the non-idealities of the timekeeper. The final experiments will show a best-case packet loss decrease from 91 % to 4 %, with a 78 % reduction in energy wasted on idle listening on the receiving side.

Chapter 2

Motivation

Our goal is to enable, for the first time, reliable, distributed wireless communication for batteryless sensors when energy conditions (harvested and/or dissipated) are so harsh that a relatively high number of power failures, in the order of dozens per second, is observed.

Batteryless ambient-powered communication will enable truly sustainable wireless networks that ideally will require *zero* maintenance. These networked intermittently-powered computers can enable applications such as on-body wearable health supervision, energy-free environment monitoring and ubiquitous sensing. While there are numerous potential applications, the intermittency inherent in these devices causes difficulty in aligning the active times of potentially communicative sensors. Devices must guess, or broadcast and receive opportunistically, hoping to stumble upon another concurrently broadcasting node. This rarely works, as the time it takes to harvest the energy required to communicate (even with backscatter methods) is one or two orders of magnitude larger than the time spent listening or sending a packet. Additionally, at the power failure, the device loses all its volatile state, including stack pointer, content of SRAM (the stack) and registers, and peripherals, including timers, become obviously not available. Without a reliable way to keep time across power failures, synchronization, and therefore message passing between two nodes, becomes opportunistic, sometimes impossible.

The severity of the problem is demonstrated through an empirical example. The rest of this Chapter serves as a motivation for a better power-failure-resilient timekeeping mechanism, and for a network synchronization method leveraging the latter.

2.1 Importance of Timekeeping

To demonstrate the problem of wake-up alignment for message passing between two intermittently-powered, battery-free sensor tags, a prelimin-

d [cm]	ω_{tx} [Hz]	ω_{rx} [Hz]	PL [%]
30	37.55	43.35	91.54
40	31.76	37.57	94.10
60	25.18	28.53	96.60

Table 2.1: Example packet loss (PL), experienced by a receiving node, for non-synchronized communication powered by an RF source positioned at distances d from transmitter and receiver. ω_{tx} and ω_{rx} represent the power failure rate of transmitter and receiver, respectively. ω_{tx} also corresponds to the number of packets sent per second (one packet per power cycle).

ary experiment was conducted. Two low-power/low-cost active radios (TI CC1101 [41]) were set to communicate in one direction, from transmitter to receiver, without acknowledgments. Each radio was controlled by a low-power MCU (TI MSP430FR5994 [43]) providing the logic to the transceiver and generating data to transmit. The transmitter was instructed to send one 4-byte packet per power cycle (the energy requirements of active radios did not allow for more). An RF signal generator was providing wireless power, and an energy harvester on both nodes was charging a buffering capacitor. The complete setup is the same used for the final evaluation and is detailed in Section 4.1. Both nodes were placed at three non-obstructed line-of-sight distances from the power generator: 30, 40 and 60 cm. By varying the distance from the power source we induced different power failure rates (at longer distances from the RF source more time is required to charge the capacitor) and different radio duty cycles. It was made sure that, in case of operation under continuous power, no packet would be lost due to channel-related reasons, like interference or topology.

The results presenting measured packet loss are given in Table 2.1. In general, the receiver consumes more energy than the transmitter due to idle listening, resulting in a higher power failure rate. The outcome of the experiment shows that, even under favorable energy conditions (at 30 cm from the RF generator), the packet loss exceeds 90 %, whereas more than 19 packets out of 20 sent are lost at distances where the incoming power is lower. This demonstrates that the opportunistic, greedy usage of energy to transmit packets under intermittent power will rarely be worth the effort. Without reliable timekeeping and a failure-resilient synchronization protocol (tuned for ultra-constrained sensing devices), message passing will not work.

2.2 Timekeeping with Transient Power

Typically, continuously-powered wireless network nodes would synchronize their transmission and reception schedules, for synchronous or asynchronous

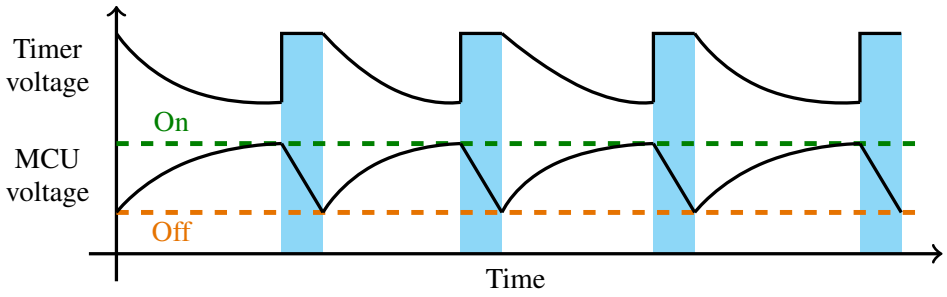


Figure 2.1: Capacitive timekeeper operation. The voltage decay of the timekeeper continues while the MCU is off. Upon reboot, the voltage on the timekeeper is sampled and mapped to elapsed time.

communication, with the help of on-board digital timers. Clearly, this solution does not apply to transiently-powered nodes, since their intermittent behavior makes digital timers unusable when the node is powered off. Thus, a different solution must be devised, keeping in mind that such solution should achieve a reasonable accuracy with an ultra-low-power consumption.

2.2.1 Timekeeping Methods

When on-board timers cannot be used, an external component has to be employed. Among known external timers are real-time clocks (RTC). RTCs, though, are typically low-granularity clocks, providing second-resolution, and consume hundreds of µA to achieve good accuracy [29]. External higher-resolution clocks, like the ones embedded on the MCU, would yield a more suitable granularity, but their power consumption would be comparable to that of the MCU itself. Capacitive timekeepers, though not well explored thus far, fit better into the low-power requirements of our target nodes.

2.2.2 Capacitive Timekeepers

A partial solution follows from the idea of *remanence timekeepers* [15, 16]. A remanence timekeeper exploits physical decay properties of an RC circuit to continuously keep track of time when there is no energy available to power a digital timer, as shown in Figure 2.1. The timekeeper charges a storage element (capacitor) and maintains the charge level while the MCU is active. While inactive, the charge decays. When activity resumes, the charge level is read and mapped to an elapsed time. The simple timekeeping circuit is presented in Figure 2.2. Remanence timekeepers suffer from two main drawbacks: (i) they are inaccurate and imprecise, and (ii) they do not maintain accuracy at the short (milliseconds) and long (minutes) timescales. Furthermore, current remanence timekeepers do not account for the active time, making it necessary to use them in combination with digital timers to

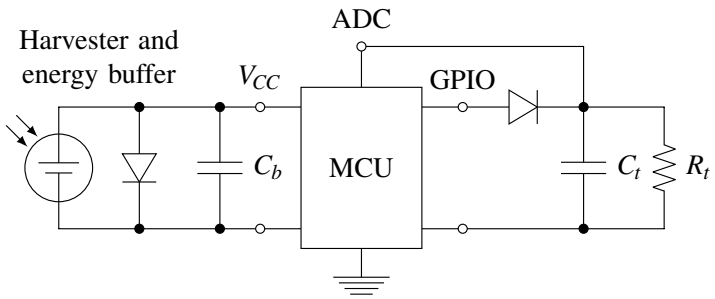


Figure 2.2: The circuit of the standard capacitive timekeeper. C_b stores the harvested energy, while C_t is the timekeeper capacitor.

keep track of time while the MCU is on. This motivates for a *completely new* timekeeping architecture—hardware and timekeeping protocol—to address requirements of accuracy, precision and flexibility.

Accuracy

The main accuracy problem current remanence timekeepers face is due to their naive software implementation. In fact, elapsed time is estimated by just using an equation and measured values for the components of the RC circuit. Inherent non-idealities of the circuit are not accounted, as well as parasitic effects caused by other components (i.e., ADC, diode and MCU).

Precision

The unaccounted variables listed above are also the cause of bad precision across different nodes. Even if all the nodes have the same components, their electrical characteristics will differ from instance to instance due to production imperfections or other external factors. The new timekeeping architecture should have a time-versus-voltage map that is tailored to the specific underlying circuit, to achieve a results that is more accurate and more consistent across multiple nodes.

Flexibility

Due to the physical nature of capacitive elements, and because an ADC can only produce a finite set of values, different capacitors are suitable for different time ranges. Small capacitor work better for short time intervals, while large capacitors suffer from low accuracy for short intervals but can capture longer periods. Our intention is to combine the properties of multiple capacitors to extend the range of time intervals that can be captured.

2.3 Synchronization with Transient Power

Low-power timekeeping solutions, like the one proposed in this work, will never achieve accuracy and resolution of energy-hungry digital clocks. Aligning radio activity of devices with such timekeepers is an unexplored problem. Compared to synchronizing the clock of multiple continuously-powered embedded nodes, this has different obstacles. The error is of another nature, is less stable and has a higher magnitude, hence requires a dedicated correction mechanism. The challenge is amplified by the fact that, due to timing boundaries imposed by energy harvesting patterns, active and inactive times cannot be arbitrarily controlled. In conclusion, an ad-hoc synchronization protocol is required to enhance communication performances, while at the same time reduce wasted energy, which could be used by the device to perform other concurrent tasks. Finally, it is important to say that, even though we decided to experiment with active radios, opportunistic intermittently-powered communication would have similar packet loss when backscatter, or any other type of asynchronous communication, is employed.

Chapter 3

Design and Implementation

As motivated in Chapter 2, a new timekeeping architecture and an ad-hoc synchronization protocol for intermittently-powered wireless network nodes must be developed. In the following Sections, design choices and implementation details will be given for both system components.

3.1 Choice of Active Radio

We target *any* active or passive radio platform. The overarching synchronization architecture is meant to be general and applicable to diverse scenarios, with the main goals of enabling timekeeping and wake-up alignment for any low-power wireless node. Choosing to use active radios to evaluate our solution has a twofold advantage. First, setting up the hardware is easier and the communication range is wider with respect to backscatter communication. Second, a higher power consumption imposes stricter timing requirements on our system, which will thus be tested in a harsher condition. Given the RF harvesting setup chosen for evaluation, we observed that two nodes can consistently exchange only one small packet per power cycle, whose size was fixed to 4 bytes to guarantee transmission in the worst-case scenario.

3.2 Hierarchical Remanence Timekeeper

A remanence timekeeper is able to measure time intervals by storing a charge on a small capacitor, when the microcontroller has power, and letting the capacitor decay through a large resistor afterwards. Then, upon a reboot, the voltage level across the capacitor will give an indication of the time elapsed since its last recharge. Ideally, it would be enough to have a resistor R, a capacitor C and an ADC to sample the capacitor's voltage V, and the equation of the discharge of an RC circuit could be used to estimate the elapsed time:

$$t = -RC \ \ln\left(V/V_0\right). \tag{3.1}$$

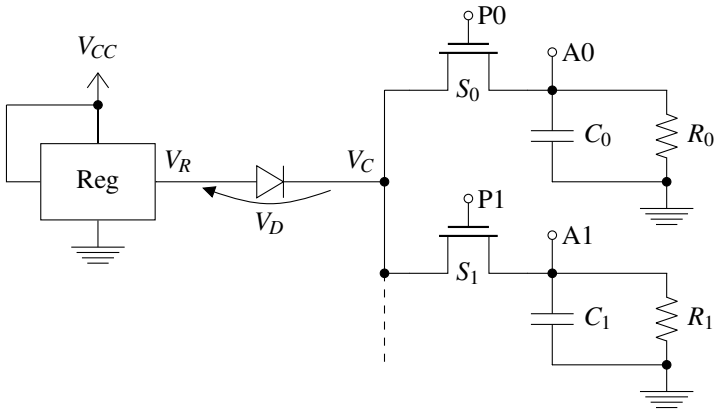


Figure 3.1: HRT circuit (only two tiers shown). The terminal Px is an output from the MCU, which controls the switch, whereas Ax is an analog input to the MCU's ADC, used to sample the voltage across the capacitor.

In actuality, capacitance and resistance never match their nominal values and other parasitic capacitors and resistors are spread through the circuit. As already mentioned in Section 2.2.2, the time-versus-voltage map should be tailored to the specific instance of the timekeeping circuit. A software calibration routine, to perform before deployment, is devised and implemented, aiming for a much better precision and accuracy of the timekeeper.

Furthermore, it has to be noted that different capacitor sizes allow different accuracy and time ranges, due to the discrete, finite set of values an ADC can produce. The accuracy is better when the discharge curve is steeper, suggesting that a smaller capacitor should yield more accurate results. On the other hand, the curve becomes less steep after two or three RC time constants, turning into a nearly-flat curve after $5RC$. This suggests that larger capacitors are better for timing longer intervals. To adapt to different needs, we propose a novel timekeeping architecture, called *hierarchical remanence timekeeper* (HRT). The HRT features an array of capacitors of different sizes to be used throughout different time ranges. The size of each capacitor, as well as the length of the array, is a design choice that the developer has to make depending on application and area constraints. Each capacitor is referred to as one *tier* of the HRT.

A further difference between the HRT and the timekeeping system used in Mayfly [15] is that, instead of only timing the inactive (off) period, the whole active-inactive power cycle is captured by the timekeeper. Compared to the hybrid solution (remanence timekeeper for the inactive time and digital timer for the active time), the HRT makes it easier to measure active times, as using a digital timer would require periodic checkpoints of the timer value into non-volatile memory, at the cost of some energy and time overhead, which increases when a higher resolution of timer checkpoints is needed.

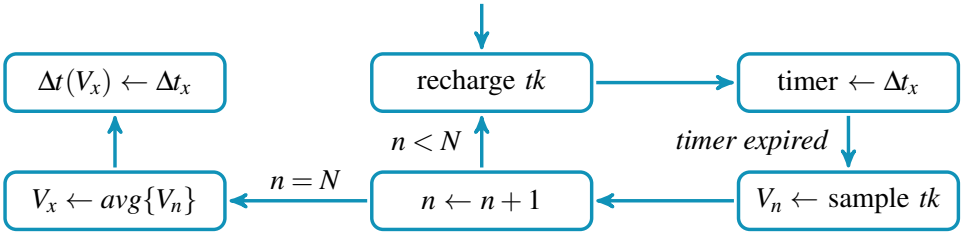


Figure 3.2: State machine of the calibration routine, where *tk* stands for timekeeper. The whole procedure is repeated for a range of several Δt_x.

3.2.1 Circuit

A schematic of the HRT circuit is provided in Figure 3.1 (for simplicity of representation, only two tiers are shown). Each tier of the HRT features a voltage regulator and a diode, shared among all tiers, and a switch (S_x), a capacitor (C_x) and a resistor (R_x), private to each tier. The regulator provides a stable voltage (2.5 V) to limit the charging of the capacitor. The diode prevents current from flowing from the capacitor back to the MCU. Finally, the switch allows the MCU to recharge the timekeeper and then open the path to let the capacitor discharge through the resistor. The complete timekeeper is very simple: one active component and one passive component for the whole HRT (regulator and diode), plus one active component and two passive components (switch, capacitor and resistor) per tier of the HRT. For instance, the total cost of the circuit does not exceed €2 for a 3-tier HRT (excluding fabrication costs for the PCB).

3.2.2 Calibration

In order to map the voltage across the capacitor to a time interval, for a specific instance of a timekeeping circuit, a software calibration tool was designed. The goal is to create a lookup table of time versus voltage, regardless of the values of R and C and of the unknown effects of the other electronics. As depicted in Figure 3.2, the calibration routine performs the following high level steps:

(1) the timekeeper is charged;
(2) a timer is set to fire after Δt_x;
(3) when the timer expires, the timekeeper is sampled and a voltage V_n is produced;
(4) steps (1) to (3) are repeated N times to produce an average voltage V_x to map to Δt_x;
(5) the whole procedure is repeated for multiple values of Δt_x.

The calibration must be performed only once, before deployment, for each tier of HRT, specifying minimum and maximum Δt_x to calibrate for.

3.2.3 Timekeeping Range Heuristics

To use one capacitor from the HRT it is sufficient to charge, let decay and sample the capacitor itself. Selecting which capacitor to use when, at run-time, is a design decision. To guide the placement of time boundaries between two adjacent capacitors in the HRT, the following model can be used. Suppose we want to place a boundary between the two smallest capacitors in the array, C_0 and C_1, with $C_0 < C_1$, to pick which one to use at run-time, based on the last measured time interval. Assume C_0 was calibrated with a resolution of δt, using an N-bit ADC and a resistor R, and during calibration and usage the capacitor is charged at V_0 (which is also the maximum value the ADC can measure). We want to find the maximum time interval $\Delta t^{\max}$ such that, for all $\Delta t_x < \Delta t^{\max}$ and $\Delta t_y = \Delta t_x + \delta t \leq \Delta t^{\max}$, the difference between the ADC values corresponding to Δt_x and Δt_y is at least K integers. Larger values of K yield better robustness against noise, but reduce the timekeeping range of a capacitor. Assume that V_x is the voltage associated to Δt_x, and V_y is associated to Δt_y ($\Delta t_x < \Delta t_y \Rightarrow V_x > V_y$). Then, we want

$$V_x - V_y \geq K \frac{V_0}{2^N}, \tag{3.2}$$

and, by applying (3.1) to (3.2), we obtain

$$\Delta t_y \leq RC_0 \ \ln\left(\frac{2^N}{K}\left(\exp\left(\frac{\delta t}{RC_0}\right) - 1\right)\right) \triangleq \Delta t^{\max}. \tag{3.3}$$

Equation (3.3) can be used to place a heuristic boundary for when to use C_0 and when C_1. Then, the same process can help place boundaries between any two adjacent capacitors in the HRT array, as well as choose how many tiers to use.

3.3 Time Synchronization

Remanence timekeepers are handy but not as accurate as digital timers. In addition, transmission and reception rates are bounded by energy availability. For these reasons, we designed an ad-hoc time synchronization protocol which is resilient against (i) timekeeping inaccuracies and (ii) energy fluctuations. Two sub-protocols are presented: *greedy transmission/delayed reception*, more simple and suitable for stable incoming harvested power, and *delayed transmission/delayed reception*, more strict but more robust against variable incoming power.

3.3.1 Analytical Model

Before describing the proposed synchronization algorithm, some preliminary concepts and definitions are introduced.

Duty Cycling

When studying wireless communication for intermittently-powered devices, it is important to notice that active and inactive periods are bounded by energy availability. In particular, the active period T_{on} has an *upper bound* $T_{\text{on}}^{\max}$, meaning that the device, and the radio, cannot be on for an arbitrarily long interval. On the other hand, the inactive period T_{off} has a *lower bound* $T_{\text{off}}^{\min}$, meaning that the recharge time cannot be arbitrarily short. These shortcomings pose strict boundaries on synchronization techniques and packet transmission, since $T_{\text{on}}^{\max}$ and $T_{\text{off}}^{\min}$ depend on several factors (harvesting circuit, environment, super-capacitor size, on/off voltage thresholds).

Wake-Up Period

The wake-up period τ is the reciprocal of the frequency a device can wake up at, and it is strictly linked to, and bounded by, $T_{\text{on}}^{\max}$ and $T_{\text{off}}^{\min}$. Again, $T_{\text{on}}^{\max}$ and $T_{\text{off}}^{\min}$ are not fixed over time, because of the factors they depend on, thus it is more proper to express those, as well as the wake-up period, as functions of time $T_{\text{on}}^{\max}(t)$, $T_{\text{off}}^{\min}(t)$ and $\tau(t)$. The wake-up period is directly influencing the contact rate, i.e., how frequently two nodes can contact each other. Our system should be adaptable to any radio and harvesting circuit, and should not make any assumption about the number of packets that can be exchanged during one connection interval.

Timekeeping Error

In general, a real time interval Δt will result in a measured time

$$\widetilde{\Delta t} = \Delta t + \varepsilon \tag{3.4}$$

where ε represents the error. Notice that ε can be positive or negative, and it is quite challenging to derive exact values for its bounds $\varepsilon^{\max}$ and $\varepsilon^{\min}$.

Listening Time

For a packet to be successfully delivered over a link between two nodes, some conditions have to hold. First, the receiver has to be able to listen, i.e., keep its radio active, for at least as long as the duration of the packet transmission. The latter will be taken for granted for the following analysis, since that concerns harvesting front end and radio, not our target problem. Furthermore, the receiver has to start listening before the first preamble bit is sent by the transmitter. If the transmitter starts sending the packet at t_{tx}, the receiver has to turn its radio on at a time t_{rx} preceding t_{tx} due to the clock inaccuracies mentioned above:

$$t_{\text{rx}} \leq t_{\text{tx}} - 2\varepsilon^{\max}, \tag{3.5}$$

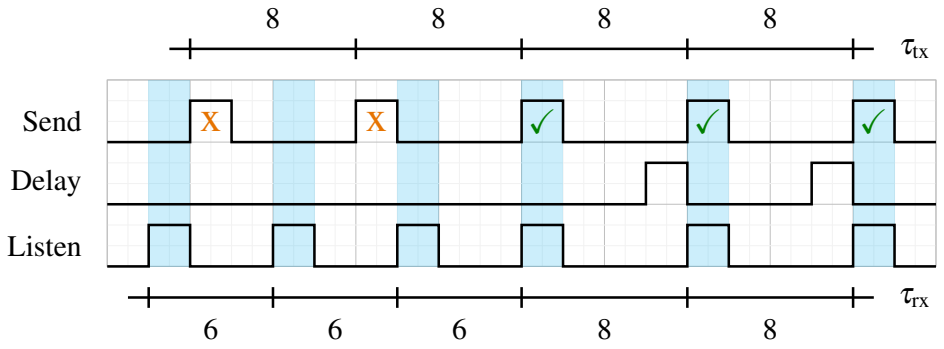


Figure 3.3: GTDR synchronization algorithm. The receiver adapts to the transmitter's wake-up period by adding delays to its reception rate.

where $2\varepsilon^{\max}$ accounts for errors on both transmitting and receiving side. Equation (3.5) gives the most strict condition, but a packet might be received even with a more delayed t_{rx}, depending on real-time clock errors.

3.3.2 Greedy Transmission/Delayed Reception (GTDR)

When the energy availability is stable due to (nearly) constant harvesting conditions, the wake-up period $\tau(k)$ (wake-up period measured at kth power cycle) is also somewhat stable. In this case, transmitter and receiver can be synchronized using a simple routine. Before sending packets, the transmitter calculates, using its on-board timekeeper, the average wake-up period as:

$$\langle \tau_{\text{tx}} \rangle_i = \frac{1}{N} \sum_{n=1}^{N} \tau_{\text{tx}}(i-n). \tag{3.6}$$

Then, the transmitter starts sending packets containing $\langle \tau_{\text{tx}} \rangle$, as well as any other data, at a rate that is approximately the reciprocal of $\langle \tau_{\text{tx}} \rangle$. When the receiver gets the first packet, it tries to schedule its next listening time after

$$\Delta_s(j) = \Delta_s(j-1) + \langle \tau_{\text{tx}} \rangle_i - \tau_{\text{rx}}(j-1) - 2\varepsilon^{\max}, \tag{3.7}$$

where Δ_s is a delay interval during which the node is put into sleep mode. In case the receiver cannot catch up with the transmitter's wake-up period due to scarcer energy availability, i.e., when Δ_s is negative, it will defer reception to the next transmission by increasing Δ_s by $\langle \tau_{\text{tx}} \rangle$, and so on until Δ_s becomes positive (or null).

Because there are no imposed restrictions on the transmission rate, and the receiver adapts to that by introducing delays, we name this synchronization scheme *greedy transmission/delayed reception* (GTDR). Figure 3.3 visualizes GTDR (in the example, the error $\varepsilon^{\max}$ is assumed to be null for

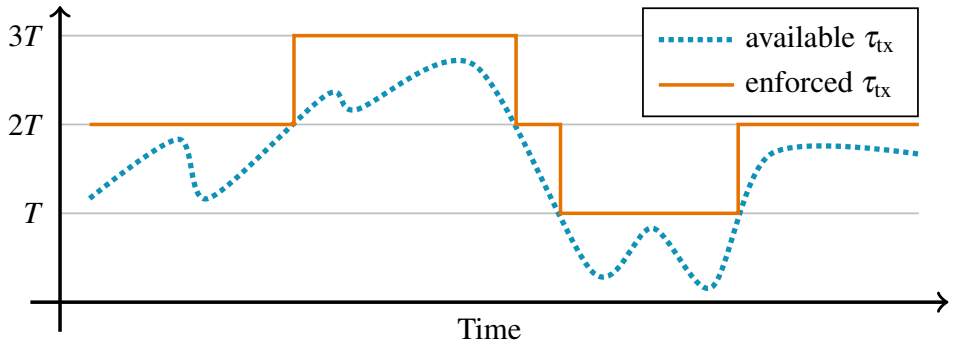


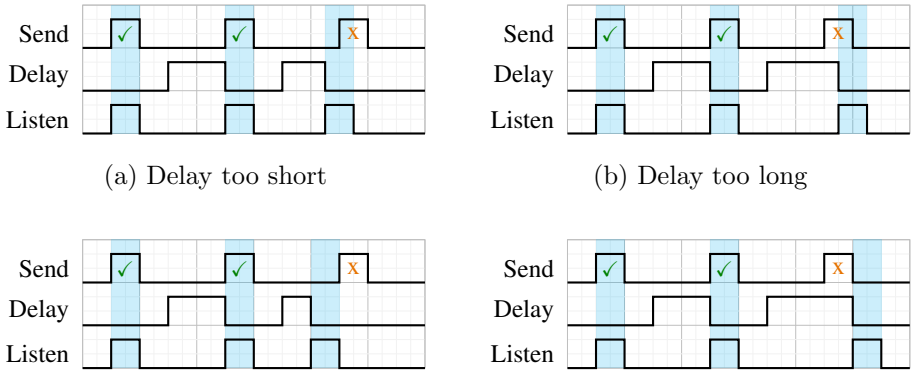
Figure 3.4: Effect of the DTDR synchronization algorithm on the transmission period. The actual period snaps to the smallest multiple of the base period T permitted by the current energy availability.

simplicity). The first packet is received at the 4th power cycle of the receiver, with a null delay Δ_s. At that point, the wake-up period of the receiver counts 6 units, and the one of the transmitter counts 8 units. This means that, for the next power cycle, the receiver will have to introduce a delay of 2 units to align to the transmitter's wake-up period. As for the last power cycle visualized in Figure 3.3, the delay on the receiving side will remain the same as for the 5th cycle (2 units), since the latest wake-up period of the receiver is the same as the transmitter's.

3.3.3 Delayed Transmission/Delayed Reception (DTDR)

In GTDR, the receiver's wake-up period is adjusted to be a feasible multiple of the quasi-constant transmitter's wake-up period. Nevertheless, more significant variations in incoming power would make the transmitter's wake-up period sway and render the average $\langle \tau_{tx} \rangle$ meaningless. A more stable transmission period can still be enforced. Choosing a base period T, the transmitter's period can be enlarged (with sleep delays) to be a multiple of T based on the energy rate, thus, for all τ_{tx} such that $(k-1)T < \tau_{tx} \leq kT$, the enforced wake-up period can always be kT (and the sleep delay $kT - \tau_{tx}$). By choosing a large T the transmission period is more robust against stronger energy variations, at the cost of lower average transmission rates. kT is computed at every power cycle and embedded into transmitted packets, so that the receiver can apply the same listening algorithm as in GTDR.

As this algorithm enforces delays on both nodes, transmitter and receiver, it is named *delayed transmission/delayed reception* (DTDR). The effect of DTDR on the transmission period is presented in Figure 3.4. Even though the unaltered wake-up period τ_{tx} would experience quick variations, the enforced period is more stable and allows for a more consistent wake-up alignment by the receiver. Visually, the transmission periods snaps to the



(a) Delay too short

(b) Delay too long

(c) Delay much too short

(d) Delay much too long

Figure 3.5: Possible scenarios for loss of synchronization.

smallest multiple of the base period T larger than the available τ_{tx}.

3.3.4 Timekeeping Error Correction

To make the proposed synchronization algorithms robust against timekeeping inaccuracies, error correction is needed. Instead of modeling the error and feeding the model to the synchronization algorithms, we opted for a run-time error correction based on proportional feedback control. At first, the receiver conservatively chooses a large value for $\varepsilon^{\max}$, derived from the evaluation of the timekeeper itself (see Section 4.2). Then, every time a packet is received after a successful Δ_s, the difference between the start of the listening activity and the start of the actual reception is tracked using an on-board digital timer. This error estimate $\hat{\varepsilon}$ is used to compute the next sleep delay. Thus, Equation (3.7) can be rewritten as:

$$\Delta_s(j) = \Delta_s(j-1) + \langle \tau_{\text{tx}} \rangle_i - \tau_{\text{rx}}(j-1) + P\hat{\varepsilon} \tag{3.8}$$

where $P \in [0, 1]$ is a proportional coefficient for tuning the error correction.

3.3.5 Sync Loss and Recovery

Even the best error correction technique cannot be resilient against all timekeeping errors, especially when the error magnitude is comparable to the maximum listening time allowed by the energy harvesting conditions. A wrong estimation of the wake-up period on either side of the connection, transmitter or receiver, induces a wrong calculation of the delay Δ_s to add to the reception schedule. Focusing on the receiving end, we can identify four possible scenarios where the computed delay leads to a wake-up misalignment, after the reception schedule was correctly aligned to the transmission

period. As depicted in Figure 3.5, the delay produced by the synchronization algorithm can be (a) too short, (b) too long, (c) much too short or (d) much too long. By too long, or too short, we mean that the listening interval overlaps with the transmission interval but they are not aligned. In the other cases (much too long and much too short), there is no overlap between listening and transmission. Regaining synchronization can be easier than starting the whole algorithm from zero, but every case needs special handling.

Delay Too Short

In the event of a computed delay just shorter than needed (Figure 3.5a), the transmission activity starts when the receiver's radio is already on, but the latter cannot sustain operation until the end of the transmission. In this case, the error tracker described in Section 3.3.4 will do its job and treat this misalignment as an error to be corrected on the next power cycle. This indeed is the simplest case of synchronization recovery and does not need any additional procedure.

Delay Too Long and Much Too Long

In case the sleep delay results to be too long (Figures 3.5b and 3.5d), the listening activity would begin after the start of the start of the transmission. This means that the error tracker cannot infer anything because the receiver could not detect a preamble sent by the transmitter. Under these circumstances, the synchronization routine will try to partially reduce the delay by $2\delta t$, where δt is the resolution of the currently used HRT's tier, as per calibration. The term $2\delta t$ accounts for a compound error induced by the timekeeper on both transmitting and receiving side, each of which is assumed to be equal to the resolution of the timekeeper itself. In case synchronization is not regained, and after a maximum number of attempts, the algorithm will restart from a null delay.

Delay Much Too Short

When the delay is much too short (Figure 3.5c), the listening activity occurs so early that there is no overlap with the transmission. Here, the receiver sees the same situation as for when the delay is too long, that is, no preamble is detected. Since it would be hard to distinguish between this and the latter case, this scenario will be treated as the latter, but of course synchronization will not be regained. Instead, the delay will eventually become null and the algorithm start over.

Chapter 4

Evaluation

The proposed timekeeping and synchronization designs were evaluated, and their performances will be discussed below. Precisely, a full description of hardware and software setup opens this Chapter. An evaluation of typical timekeeping errors produced by the HRT follows. In conclusion, an extensive performance analysis of the synchronization algorithms is provided, including a quantification of throughput, packet loss and energy efficiency.

4.1 Experimental Setup

We conducted experiments using RF energy as for the demonstration example given in Section 2.1, using the same configuration. As said earlier, the setup was tested on continuous power to make sure that no packet would be lost due to channel-related reasons.

4.1.1 Hardware Configuration

We used a Powercast TX91501 power generator [34] to provide RF energy, emitting at 915 MHz with 1 W EIRP using a 60° directional antenna, and a Powercast P2110 energy harvester [33]. The DC current output by the harvested was used to charge a 1-mF energy buffer (a capacitor). A Schmitt trigger was cascaded to the capacitor in order to give power to the MCU when the energy buffer was charged above 3 V, and to cut the power when the voltage across the energy buffer was dropping below 1.8 V (1.2 V of hysteresis). A TI MSP430FR5994 low-power MCU [43] was controlling a TI CC1101 low-power active radio [41], configured to transmit at a rate of 76 kBaud/s with a transmission power of −30 dBm.

4.1.2 Communication Setup

The radio was instructed to send 14-byte packets, consisting of 8 bytes of preamble and sync word, 4 bytes of payload and 2 bytes of CRC (for error

detection). At each power cycle, the transmitting node was sending only one packet, as well as the receiving node was listening to receive at most one packet. Due to energy availability and requirements, it was observed that only one packet could be guaranteed to be sent in the worst-case scenario, hence we conservatively decided to always send one packet per power cycle. In a real application, the energy remaining after transmission or successful reception, if any, can be used to perform other tasks (in our evaluation the MCU cycles through an infinite loop). The amount of leftover energy is evaluated in Section 4.3.4.

4.1.3 Measurement Equipment

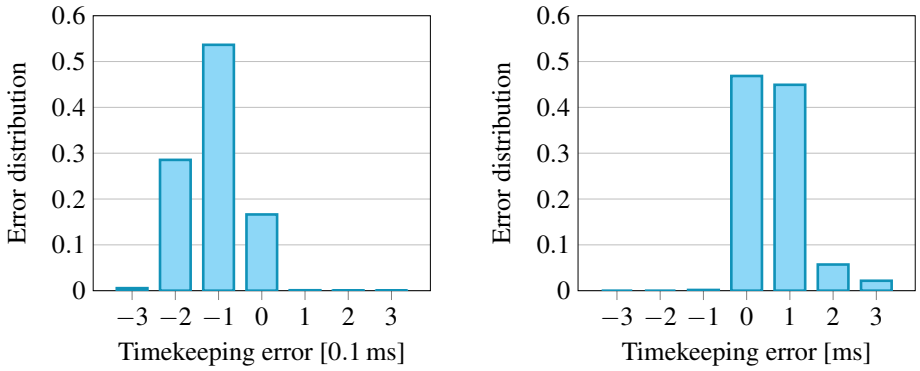
For all experiments, data gathering was conducted with the aid of a Saleae logic analyzer with ADC capabilities [38], connected to the MCU. Output pins of the microcontroller were used to generate logic traces marking insightful events, including power state of the MCU (on/off), radio activity and packet transmission and reception. As for energy measurements, we used a TI INA225 current sense amplifier [42], whose output was again captured by the logic analyzer. Logic and analog traces were parsed using multiple Python scripts to produce the results presented in this Chapter.

4.1.4 Software Libraries

The software dedicated to timekeeper and synchronization algorithms was implemented in C and wrapped into two well-documented libraries. A complete implementation of the GTDR synchronization algorithm takes 242 B of device memory, on the receiver, while DTDR takes 226 B on the transmitter. Additionally, the software library accompanying the HRT amounts to 1990 B, and the ADC lookup table occupies 8 kB of memory per tier.

4.2 Timekeeping Accuracy

As a first evaluation step, the accuracy of the HRT was measured. In particular, the experiments were conducted on two capacitors, of 22 nF and 100 nF, which will then be used during the evaluation of the intermittent communication. The smaller capacitor was calibrated and tested to capture time intervals in the range 10–50 ms, with a resolution of 0.1 ms, whilst the larger was evaluated for the range 10–200 ms, with a resolution of 1 ms. To measure timekeeping accuracy, a square wave generator was resetting the MCU at a predetermined period, to simulate energy-harvesting-induced power failures. Each capacitor was tested with a set of such periods, covering ranges and resolutions specified above, and multiple times (1000) per period. The timekeeper's estimation was logged by the MCU and then compared with the pre-generated square wave periods.



(a) Cap. 22 nF, range 10–50 ms (b) Cap. 100 nF, range 10–200 ms

Figure 4.1: Timekeeping error distribution of two capacitors.

Figure 4.1 depicts the overall error distribution measured for the experiment. As can be observed, both capacitors report a Gaussian-like error distribution with low variance. The smaller one is affected by a maximum absolute error of 0.2 ms, for the specific range, besides sporadic outliers. As for the larger one, the typical error oscillates between 0 and 1 ms. Such results are rather encouraging, and suggest that the HRT can be definitely used in applications with millisecond-accuracy requirements. Nevertheless, the presence of outliers, even one in a hundred, can be injurious for our synchronization goals. Unfortunately, at this stage, capacitive timekeepers are still more unstable than digital clocks.

Among the factors that might influence the performance of the HRT are (i) circuit interferences and (ii) temperature. The former can be stemmed by realizing a PCB to have microcontroller and timekeeping circuit steady on the same board and to reduce antenna effects of breadboard wires. Regarding the latter issue, temperature-resilient capacitive elements can be used, but the cost of the HRT would increase. The aforementioned solutions should be implemented and tested.

4.3 Intermittently-Powered Communication

For most of the wireless communication experiments, we have evaluated the performance of our solution under different energy harvesting conditions. At first, the two intermittently-powered nodes were places next to each other, and at different distances from the power generator, 30, 40 and 60 cm, representative of three different energy availability scenarios. Then, a more dynamic experiment was performed, during which the power generator was moved back and forth from the fixed nodes, to induce a higher-magnitude swing on the incoming power. Every single experiment was run continuously

Parameter	Symbol	Value	Unit
HRT tier 0 (22 nF) calibration interval	–	0–45	ms
HRT tier 0 (22 nF) calibration resolution	–	0.2	ms
HRT tier 1 (100 nF) calibration interval	–	45–200	ms
HRT tier 1 (100 nF) calibration resolution	–	1	ms
Average window size for GTDR	N	4	–
Error correction proportional coefficient	P	0.5	–
Sync recovery maximum attempts	–	5	–
Base transmission period for DTDR	T	20	ms

Table 4.1: System parameters empirical tuning.

for two minutes. To synchronize the nodes, a 2-tier HRT with two capacitors of 22 nF and 100 nF was used, which was observed to provide enough flexibility for the timing ranges involved. As a baseline for comparison, we also performed all the experiments *without* timekeeping and synchronization enabled. Moreover, some system parameters described in Chapter 3 were empirically tuned, and the values that yielded the most performing configuration are recapped in Table 4.1.

4.3.1 Evaluation Metrics

Throughout the whole study, only the topology of power generator versus harvesters was altered. At the two fixed relative distances of 30 and 40 cm, the experiment was run twice: (i) without, and (ii) with a metal obstruction between power source and receiving node, the latter to make the receiver experience a lower incoming energy rate compared to the transmitter, and hence to evaluate the situation in which the receiver should wake up for reception at a fraction of the transmission frequency. To evaluate DTDR, the relative distance between power generator and harvester was dynamically varied from 30 to 80 cm, back and forth, with a round-trip time of 20 s.

In the following Sections, multiple performance metrics will be reported on. First of all, *throughput* and *packet loss* will be discussed. Then, the energy waste linked to *idle radio listening*, on the receiving side, will be quantified. In strict relation to this, we will also present the *excess energy* for baseline and sync-enabled configurations. The latter quantifies how much energy is left after transmitting or successfully receiving one packet. In real applications, this energy can be spent by the node to perform other tasks, like sensing or processing data. Finally, in order to have a sense of the energy conditions experienced, two additional metrics are considered: *average wake-up period* and *synchronization challenge*. The average wake-up period of the transmitter actually represents the packet rate (recall, one packet per power cycle). As for the receiver, it gives an indication of the

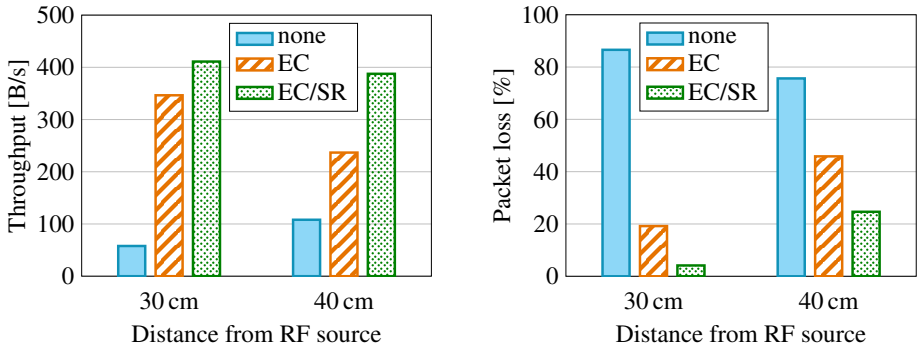
d [cm]	$\langle\tau_{\text{tx}}\rangle$ [ms]	$\langle\tau_{\text{rx}}\rangle$ [ms]	γ	sync	TP [B/s]	PL [%]
30	26.63	23.07	0.32	none	43.98	91.54
				GTDR	411.02	4.14
30 *	24.57	28.59	0.36	none	38.97	93.10
				GTDR	256.90	52.23
40	31.49	26.62	0.36	none	26.13	94.10
				GTDR	387.57	12.44
40 *	29.64	39.44	0.42	none	18.67	96.02
				GTDR	135.33	71.17
60	39.71	35.05	0.41	none	11.90	96.60
				GTDR	157.50	56.25
moving	–	–	–	none	14.00	96.10
				DTDR	32.90	87.30

Table 4.2: Throughput (TP) and packet loss (PL) for intermittent communication between two nodes, placed next to each other, at multiple fixed distances d from the power generator, plus the moving distance. The asterisk marks experiments with an obstacle between power source and receiving node, $\langle\tau_{\text{tx}}\rangle$ and $\langle\tau_{\text{rx}}\rangle$ are the average wake-up period of transmitter and receiver, respectively, and γ is the synchronization challenge.

energy availability at this node with respect to the transmitter. A receiver's wake-up period smaller than the transmitter's means that, hypothetically, the receiver has enough energy availability to catch all the transmitted packets. In the opposite case, the receiver is expected to wake up at a frequency that is a fraction of the packet rate. The synchronization challenge, denoted as γ, is an indicator of how accurate the synchronization has to be in order to cope with the short listening interval allowed by the incoming energy. It is defined as the ratio between the typical transmission interval and the longest possible listening interval. In our experiments, the transmission interval is fixed to 1.46 ms (since packet length and data rate are constant), while the longest listening interval depends on the energy availability. For instance, $\gamma = 0.25$ means that the receiver can listen for an interval four times as long as the transmission interval.

4.3.2 Throughput and Packet Loss

The first analysis focuses on throughput and packet loss. Table 4.2 summarizes the results of the experiment. Normally, the average wake-up period of the receiver, $\langle\tau_{\text{rx}}\rangle$, was shorter than the one of the transmitter, $\langle\tau_{\text{tx}}\rangle$. As discussed above, a metal obstacle was placed between the power generator and



(a) Throughput (b) Packet loss

Figure 4.2: Throughput and packet loss, at two distances from the power generator, for three flavors of GTDR: without any correction (none), with error correction (EC), with error correction and sync recovery (EC/SR).

the receiving node, for distances of 30 and 40 cm, to evaluate the opposite case as well. These experiments are marked with an asterisk in Table 4.2. It has to be considered that, when $\langle \tau_{tx} \rangle < \langle \tau_{rx} \rangle \leq 2 \langle \tau_{tx} \rangle$, as in the experiments with the obstacle, the packet loss cannot be less than 50 %, since the receiver does not have enough energy to catch two consecutive transmitted packets. Moreover, when GTDR is enabled, the average wake-up period of the receiver $\langle \tau_{rx} \rangle$ refers to those power cycles without any delay introduced, that is, the natural wake-up period dictated by the incoming energy rate. Average wake-up period and synchronization challenge are only reported once per energy configuration, since there was hardly any variation between the asynchronous experiment and the sync-enabled one. Finally, average wake-up period and synchronization challenge are not reported for the dynamic distance experiment, since the varying energy conditions result in a high variance of such metrics.

GTDR

The results concerning GTDR are rather encouraging. For the experiments at fixed distances from the power source, the baseline throughput, that is for non-synchronized communication, was improved by an average factor of 10. As for the packet loss, it was reduced from 91 % to 4 %, in the most favorable energy conditions, and from 93 % down to almost 50 % at 30 cm with the metal obstacle. Even for larger values of γ, the packet loss was reduced by 25 percentage points, at 40 cm with the obstacle, and by 40 points, at 60 cm. The results suggest that GTDR is very successful under good energy availability, and still effective when the incoming energy is less. Notice that its performance could be even better when combined with a more low-power communication techniques like backscatter, where γ would be smaller.

Consumption parameter	Value	Unit
ADC for timekeeping, per power cycle	21.3	nJ
Timekeeper recharge, per power cycle	1.48	µJ
MCU running at 8 MHz	3.83	mW
MCU in low-power mode	0.39	mW
Radio in idle state	5.1	mW
Radio in listening state	15	mW

Table 4.3: Measured consumption of different system components.

DTDR

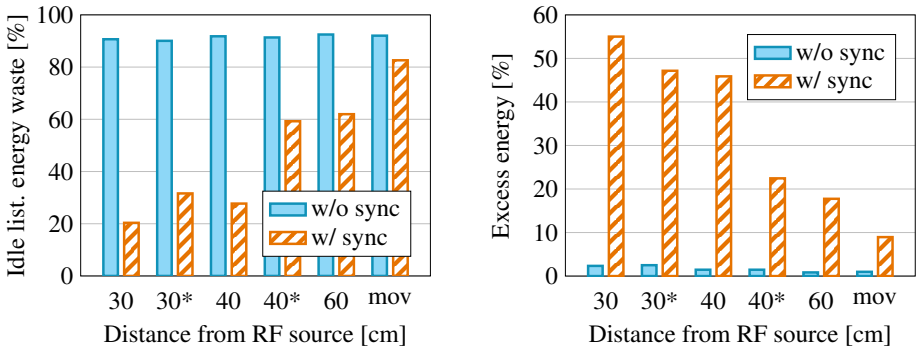
Unfortunately, DTDR was not as successful. Even though some improvement over the baseline were observed (throughput 2.35 times larger), they were not satisfactory. Since the HRT proved to be accurate enough for intermittent communication, the reason for the low performance of DTDR must lie in its implementation.

4.3.3 Error Correction and Sync Recovery

To motivate the need for error correction and sync recovery, and to demonstrate their benefit, the two experiments at 30 and 40 cm, without any obstacle, were run again (i) disabling error correction and sync recovery, and (ii) disabling sync recovery only. As can be observed in Figure 4.2, the error correction functionality is a fundamental contributor to the efficiency of GTDR. Without that, throughput and packet loss are bad at 40 cm from the power generator, and, counter-intuitively, they are even worse at 30 cm (this has to do with randomness though). As mentioned earlier, even the smallest error caused by the low-power HRT can be destructive. When error correction is introduced, performances do get better, especially in the high-energy scenario (30 cm). Sync recovery has a good impact as well, which is better appreciated in lower-energy conditions, where synchronization is lost more frequently.

4.3.4 Energy Efficiency

Throughput is extremely important in communication systems. Nevertheless, when low-power, energy-harvesting devices are involved, other metrics have to be considered as well. For network nodes that have limited energy availability, like the ones we are targeting, reducing energy waste can have significant benefits and can allow interesting applications. To quantify the energy efficiency of GTDR and DTDR, the same set of experiments as in Table 4.2 was used. For the quantitative analysis, we used power consumption facts derived from empirical measurements, summarized in Table 4.3.



(a) Idle listening energy waste

(b) Excess energy

Figure 4.3: Energy efficiency, with data normalized to the total harvested energy, at multiple fixed distances from the power generator (with GTDR), plus the moving distance (with DTDR). Experiments with an obstacle between power source and receiving node are marked with an asterisk.

Even though the MCU's V_{CC} is not constant (it follows the voltage across the energy buffering capacitor), a fixed value of 3 V was assumed for faster measurements. Moreover, the consumption of the timekeeper was approximated by a one-tier HRT, with a capacitor of 470 nF and a resistor of 1 MΩ.

Idle Listening

Idle listening refers to the receiver being in listening mode without actually receiving. Especially in the case of active radios, wasted radio activity is extremely deleterious for the system, since it is the most expensive operation. Figure 4.3a visualizes the amount of energy spent on idle listening, normalized to the total energy consumption. When no synchronization algorithm is applied, the percentage of harvested energy that is wasted for idle listening is at least 90 %, in all cases. As for synchronized communication, the energy waste can be reduced down to 20 %, in the best case (at 30 cm with GTDR), and to 82 %, in the worst case (represented by the dynamic energy scenario, where DTDR is used). As confirmed by this experiment, idle listening is a direct consequence of non-synchronized packet exchange.

Excess Energy

Energy efficiency can be also analyzed from a different angle. As described earlier, we designed our system to send or receive one packet per power cycle. The remaining energy stored in the buffer can be effectively used to perform other tasks after each successful transmission or reception. This excess energy was quantified to demonstrate that not only can our system achieve better throughput, but it can save enough energy to spend on other

operations alongside wireless communication. The amount of energy saved on the receiving side is presented in Figure 4.3b. Without synchronization, the amount of leftover usable energy, normalized to the total amount of energy gathered from the environment, lies below 2.5 %. GTDR improves the percentage of excess energy up to 55 %, when good synchronization is achieved, and the overall excess energy is always 15 to 20 times larger than for the baseline, except for the dynamic energy case (when DTDR is used). This proves that, besides achieving a higher throughput, GTDR is able to save useful energy as well. As for throughput and packet loss, error correction has a large impact on energy efficiency, since it helps minimize untimely (early) wake-ups of the receiver.

Chapter 5

Related Work

With the aim of positioning this work in its context, and providing the reader with a comparison with the state of the art, this Chapter makes a summary of known relevant papers and other literature.

5.1 Low-Power MAC Protocols

There is an enormous set of works centered around medium access control for low-power applications, targeting wireless sensors, energy harvesting networks and backscatter communication.

5.1.1 Wireless Sensor Networks

WSNs consist of low-power nodes that are equipped with sensors and wireless transceivers in order to collaboratively perceive, process and disseminate environmental data. The communication performed by the transceiver circuitry is the activity that consumes the most energy on battery-powered sensor platforms. Inefficient use of the radio leads to inefficient operation, and hence waste of energy. In order to reduce their energy consumption and extend their lifetime from days to years, sensor nodes repeatedly switch off and on, i.e., duty cycle, their radios [5]. The objective of MAC protocols targeted for WSNs is to reduce energy consumption by duty-cycling radios to eliminate wasted energy due to (i) *idle listening*, (ii) *overhearing* the messages destined for the other nodes, and (iii) *collisions* and *retransmission* [6].

The difficulty of duty-cycling is that nodes should know beforehand when packets are coming and then switch on their radios to receive the data. Coordination errors may cause packet losses and may reduce the amount of data that can be communicated. MAC schemes that allow random channel accesses, i.e., asynchronous protocols [32, 4, 40], do not require any a priori knowledge on when the sleep/wake-up cycle will take place. As an example, a sender node can send a long preamble in order to guarantee that the

receiver nodes wake up and detect a start symbol, and then receive the payload of the packet. This well-known technique is referred to as *low-power listening* [32]. On the other hand, slotted access MAC protocols [45, 9] require an explicit coordination among the sensor nodes in order to switch on and off their radios simultaneously. Nodes wake up at the beginning of each slot, they sense the communication channel and perform collision avoidance, then they send or receive packets and go back to sleep after a fixed amount of time. In frame based protocols [44], contention and collisions are prevented by assigning dedicated scheduled slots to sender and receivers. In conclusion, all prior works targeted for WSNs strive to optimize energy consumption and *assume continuous operation*, thus relying on continuous, more accurate clocks. However, our objective in this work is to utilize harvested energy in order to ensure communication among the batteryless nodes, while resorting to a more error-prone clock system.

5.1.2 Energy Harvesting Sensor Networks

MAC protocols designed for wireless sensor nodes with energy harvesting capability are more challenging, since ambient energy patterns are not stable and easily predictable [23, 39]. Conventional protocols in WSNs are not designed considering energy harvesting characteristics. Unlike those, MAC protocols for energy-harvesting nodes target the optimal use of the harvested energy to maximize throughput, instead of saving energy. Due to the fact that energy harvesting patterns of the nodes might be different, synchronous MAC-layer designs were excluded by the research community. Since a node that experiences a power failure loses its notion of time, wake-up alignment had not been investigated yet. Therefore, current protocols in the literature fall under the asynchronous category. For instance, an algorithm that adaptively adjusts sleeping periods by considering the incoming traffic and energy status was proposed [30]. Differently, other protocols regulate energy consumption to keep all nodes in the so-called *energy neutral operation* mode [11, 19], meaning that the consumed energy is always less than the harvested energy (thus power never fails). Our proposal diverges from aforementioned prior works targeted for energy harvesting systems in the way that we propose a *synchronized* duty-cycling scheme based on batteryless timekeeping, which aligns nodes' schedules *across power failures*.

5.1.3 Delay Tolerant Networks

Delay tolerant networks (DTNs) allow long and unpredictable delays during data transmission. In such networks, energy efficiency, flexible traffic loads and mobility support—despite intermittent connectivity—is the main subject of MAC protocols [51]. Techniques such as sleep scheduling and duty-cycling are also relevant in these networks [7]. However, energy harvesting

patterns of the nodes and loss of connections due to power failures are overlooked. In fact, intermittently-powered network nodes lose connection *at every reboot*, multiple times per second. Rather, DTNs generally consider intermittent and dynamic connections due to mobility, at a considerably lower rate [18]. In addition, we consider nodes with non-negligible resource constraints, as opposed to the powerful nodes in DTNs with (theoretically) infinite computation capabilities and continuous power supply.

5.1.4 Backscatter Networks

Backscatter enables ultra-low-power communication by eliminating energy-hungry active radio components, resulting in energy requirements several orders of magnitude lower than their counterparts [48, 50, 49]. Most of the backscatter networks, like the one proposed by Alevizos et al. [1], allow communication only between batteryless devices and a continuously-powered master device. Recent studies have addressed this issue and enabled communication among batteryless nodes [22, 36, 37]. They have demonstrated a multi-hop backscatter network that can relay information from one physical point to another. Nevertheless, the MAC-layer designs proposed in these works ignore power failures during communication. Moreover, they mostly consider communication between the reader and a single tag. Instead, our solution is intended to synchronize nodes that make use of any physical layer, despite power failures, and can be as well employed in backscatter networks to remove reader coordination during communication.

5.2 Network Time Synchronization

WSN nodes' clocks are generally sourced by cheap oscillators that are prone to significant, unpredictable instabilities. Due to these drifts, the clock of each node degrades and diverges over time. Consequently, it is mandatory to perform periodical time synchronization to ensure that nodes are able to acquire a common notion of time and perform coordinated actions [26]. For instance, as in our case, the synchronized software clock can be used to coordinate duty-cycling in MAC protocols.

There is a considerable amount of protocols dedicated to WSNs time synchronization [10, 28, 13, 20, 12, 21, 47]. The general goal of these works is to build a network-wide notion to share with all the nodes. Whenever a fresh time information is received by a node, the offset of the local clock is updated to minimize the estimated error. As an example, in the *Flooding Time Synchronization Protocol* [28], each node receives flooded time information generated by a dedicated reference node, and performs least-squares regression on the received value to update its local clock. Similarly, as proposed by Yıldırım et al. [47], proportional-integral controllers can be used to compensate for clock drifts and offsets. All these works though do not

consider frequent power failures and, in turn, the loss of synchronization state—a common phenomenon among batteryless platforms.

An initial attempt to time synchronization for batteryless RFID systems has been done [46]. However, this work considers synchronization between an RFID reader and a sensor tag, the latter using the built-in clock hardware of the microcontroller that reset upon power failures. Rather tan relying on the volatile timers of the microcontroller, our work employs an external timekeeping component. Moreover, instead of using digital clock readings, our HRT transduces energy into discrete time values.

5.3 Batteryless Timekeeping

In order to keep track of time in batteryless platforms, across power failures, two novel timekeeping techniques were proposed by Hester et al. [16]. Both approaches, named *TARDIS* and *CusTARD*, exploit the decay of physical components, SRAM and capacitor respectively, to generate a continuous notion of time resilient to power failures. The former solution is, albeit ingenious, difficult to handle, and yields very poor accuracy. CusTARD, the idea that is also a foundation of Mayfly [15], monitors the voltage drop across a capacitor, that decays while the microcontroller is off, to estimate the elapsed time. CusTARD is also the forefather of our hierarchical remanence timekeeper. The issues related to its primordial implementation have been discussed through Chapters 2 and 3.

Chapter 6

Discussion

Although what has been presented so far has the potential to be a fundamental step towards reliable energy-harvesting IoT networks, it is still in its infancy. In an effort to help future developments of intermittent communications, known limitations of the presented prototype, as well as proposed next directions, are briefly discussed below.

6.1 Observed Limitations

As witnessed by the evaluation in Chapter 4, the proposed system has some limitations in terms of timekeeping and synchronization under highly-varying energy conditions. The HRT is a great leap towards zero-energy timekeeping, but its reliability has not been tested enough, particularly in different environmental conditions (e.g., varying temperature). Moreover, it has not been investigated whether the variance of its output is due to circuit interferences (caused by low-quality breadboard interconnects) or inherent limitations of the design. Regarding synchronization, GTDR struggles to achieve an efficient wake-up alignment of two intermittently-powered nodes when the synchronization challenge γ is high. Although there is room for improvement, this issue could be a fundamental limitation of networks featuring active radios. On the other hand, DTDR yielded poor results under highly-dynamic energy conditions. We believe that better tuning of the algorithm could result in better performances.

6.2 Future Work

In order to make this research complete, and its outcome usable in real systems, further steps have to be taken. We envision a number of directions that could be explored in the future. To begin, the HRT should be tested over a range of environment temperatures, in order to characterize its stability with respect to that. Furthermore, it would be of great value to realize a

PCB integrating the whole timekeeping circuit, to verify if that can improve the precision of its time estimations. For what concerns synchronization, GTDR has proven to yield very good performance when the synchronization challenge γ does not exceed 0.4 (cf. Table 4.2). For this reason, it would be interesting to test our system with a lower-power physical-layer protocol, like backscatter, and measure its efficiency in that case. In addition to that, DTDR should go through more development iterations and more accurate tuning. Given the good timekeeping results the HRT can achieve, synchronization under dynamic incoming energy should be possible. On a higher level, the designed synchronization algorithm should be embedded in a full MAC-layer protocol and employed to develop unprecedented energy-harvesting applications, and finally render the battery-free IoT possible.

Chapter 7

Conclusions

In this thesis, the feasibility of practical batteryless intermittently-powered wireless communication was demonstrated by introducing a novel timekeeping architecture and an accompanying synchronization protocol. The ultra-low-power timekeeper, denoted as *hierarchical remanence timekeeper* (HRT), uses fundamental properties of an RC circuit to capture and keep track of time even when the microcontroller is powered off. It allows a wide range of time intervals to be measured with up to sub-millisecond accuracy, using only a handful of off-the-shelf components. The HRT was then used to perform wake-up alignment of intermittently-powered network nodes. The *greedy transmission/delayed reception* synchronization algorithm is able to reduce packet loss, under good energy harvesting conditions, from 91 % down to 4 %, while reducing energy waste by 78 %, compared to an asynchronous communication scheme. With less stable incoming energy, the *delayed transmission/delayed reception* algorithm reduces packet loss and energy consumption, and improves throughput, but not to a satisfactory level (the throughput is 2.35 times larger than the asynchronous baseline). In conclusion, we can confirm that intermittent, battery-free wireless communication is a real possibility, provided that some extra work will be carried out to refine the performances of capacitive timekeeper and intermittent synchronization algorithms.

Bibliography

[1] P. N. Alevizos, K. Tountas, and A. Bletsas. Multistatic Scatter Radio Sensor Networks for Extended Coverage. *IEEE Transactions on Wireless Communications*, 2018.

[2] D. Balsamo, A. S. Weddell, A. Das, A. R. Arreola, D. Brunelli, B. M. Al-Hashimi, G. V. Merrett, and L. Benini. Hibernus++: A Self-Calibrating and Adaptive System for Transiently-Powered Embedded Devices. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(12):1968–1980, 2016.

[3] N. A. Bhatti and L. Mottola. HarvOS: Efficient Code Instrumentation for Transiently-Powered Embedded Sensing. In *2017 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 209–220, April 2017.

[4] M. Buettner, G. V. Yee, E. Anderson, and R. Han. X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 307–320. ACM, 2006.

[5] N. Burri, P. Von Rickenbach, and R. Wattenhofer. Dozer: Ultra-Low Power Data Gathering in Sensor Networks. In *Proceedings of the 6th international conference on Information processing in sensor networks*, pages 450–459. ACM, 2007.

[6] R. C. Carrano, D. G. Passos, L. C. S. Magalhães, and N. Célio Vinicius. Survey and Taxonomy of Duty Cycling Mechanisms in Wireless Sensor Networks. *IEEE Communications Surveys and Tutorials*, 16(1):181–194, 2014.

[7] B. J. Choi and X. Shen. Adaptive Asynchronous Sleep Scheduling Protocols for Delay Tolerant Networks. *IEEE Transactions on Mobile Computing*, 10(9):1283–1296, 2011.

[8] A. Colin and B. Lucia. Chain: Tasks and Channels for Reliable Intermittent Programs. In *Proceedings of the 2016 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*, OOPSLA 2016, pages 514–530, New York, NY, USA, 2016. ACM.

[9] A. El-Hoiydi and J.-D. Decotignie. WiseMAC: An Ultra Low Power MAC Protocol for Multi-hop Wireless Sensor Networks. In S. E. Nikoletseas and J. D. P. Rolim, editors, *Algorithmic Aspects of Wireless Sensor Networks*, pages 18–31, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[10] J. Elson, L. Girod, and D. Estrin. Fine-Grained Network Time Synchronization Using Reference Broadcasts. *SIGOPS Oper. Syst. Rev.*, 36(SI):147–163, Dec. 2002.

[11] X. Fafoutis and N. Dragoni. ODMAC: An On-Demand MAC Protocol for Energy Harvesting-Wireless Sensor Networks. In *Proceedings of the 8th ACM*

Symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks, pages 49–56. ACM, 2011.

[12] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. Efficient Network Flooding and Time Synchronization with Glossy. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pages 73–84. IEEE, 2011.

[13] S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-Sync Protocol for Sensor Networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 138–149. ACM, 2003.

[14] J. Hester, T. Scott, and J. Sorber. Ekho: Realistic and Repeatable Experimentation for Tiny Energy-harvesting Sensors. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, SenSys '14, pages 330–331. ACM, 2014.

[15] J. Hester, K. Storer, and J. Sorber. Timely Execution on Intermittently Powered Batteryless Sensors. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, SenSys '17, pages 17:1–17:13. ACM, 2017.

[16] J. Hester, N. Tobias, A. Rahmati, L. Sitanayah, D. Holcomb, K. Fu, W. P. Burleson, and J. Sorber. Persistent Clocks for Batteryless Sensing Devices. *ACM Trans. Embed. Comput. Syst.*, 15(4):77:1–77:28, Aug. 2016.

[17] M. Hicks. Clank: Architectural Support for Intermittent Computation. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, pages 228–240, New York, NY, USA, 2017. ACM.

[18] S. T. Kouyoumdjieva and G. Karlsson. Impact of Duty Cycling on Opportunistic Communication. *IEEE Transactions on Mobile Computing*, 15(7):1686–1698, 2016.

[19] T. N. Le, A. Pegatoquet, O. Berder, and O. Sentieys. Energy-Efficient Power Manager and MAC Protocol for Multi-Hop Wireless Sensor Networks Powered by Periodic Energy Harvesting Sources. *IEEE Sensors Journal*, 15(12):7208–7220, Dec 2015.

[20] C. Lenzen, P. Sommer, and R. Wattenhofer. Pulsesync: An Efficient and Scalable Clock Synchronization Protocol. *IEEE/ACM Transactions on Networking (TON)*, 23(3):717–727, 2015.

[21] R. Lim, B. Maag, and L. Thiele. Time-of-Flight Aware Time Synchronization for Wireless Embedded Systems. In *EWSN*, pages 149–158, 2016.

[22] V. Liu, A. Parks, V. Talla, S. Gollakota, D. Wetherall, and J. R. Smith. Ambient Backscatter: Wireless Communication out of Thin Air. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 39–50. ACM, 2013.

[23] X. Lu, P. Wang, D. Niyato, D. I. Kim, and Z. Han. Wireless Networks with RF Energy Harvesting: A Contemporary Survey. *IEEE Communications Surveys & Tutorials*, 17(2):757–789, 2015.

[24] B. Lucia, V. Balaji, A. Colin, K. Maeng, and E. Ruppel. Intermittent Computing: Challenges and Opportunities. In *2nd Summit on Advances in Programming Languages (SNAPL 2017)*, Leibniz International Proceedings in Informatics (LIPIcs), pages 8:1–8:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[25] B. Lucia and B. Ransford. A Simpler, Safer Programming and Execution Model for Intermittent Systems. In *Proceedings of the 36th ACM SIGPLAN*

Conference on Programming Language Design and Implementation, PLDI '15, pages 575–585, New York, NY, USA, 2015. ACM.

[26] J. Ma, W. Lou, Y. Wu, X.-Y. Li, and G. Chen. Energy Efficient TDMA Sleep Scheduling in Wireless Sensor Networks. In *INFOCOM 2009, IEEE*, pages 630–638. IEEE, 2009.

[27] K. Maeng, A. Colin, and B. Lucia. Alpaca: Intermittent Execution Without Checkpoints. *Proc. ACM Program. Lang.*, 1(OOPSLA):96:1–96:30, Oct. 2017.

[28] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi. The Flooding Time Synchronization Protocol. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 39–49, New York, NY, USA, 2004. ACM.

[29] Maxim Integrated. DS3231 RTC. `datasheets.maximintegrated.com/en/ds/DS3231.pdf`. Last accessed: Sep. 2018.

[30] T. D. Nguyen, J. Y. Khan, and D. T. Ngo. An Adaptive MAC Protocol for RF Energy Harvesting Wireless Sensor Networks. In *Global Communications Conference (GLOBECOM), 2016 IEEE*, pages 1–6. IEEE, 2016.

[31] M. R. Palacín and A. de Guibert. Why Do Batteries Fail? *Science*, 351(6273), 2016.

[32] J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107. ACM, 2004.

[33] Powercast Co. P2110 Powerharvester Evaluation Board. `www.powercastco.com/wp-content/uploads/2016/11/p2110-evb1.pdf`. Last accessed: Aug. 2018.

[34] Powercast Co. TX91501 Powercaster Transmitter. `www.powercastco.com/wp-content/uploads/2016/11/User-Manual-TX-915-01-Rev-A-4.pdf`. Last accessed: Aug. 2018.

[35] B. Ransford, J. Sorber, and K. Fu. Mementos: System Support for Long-running Computation on RFID-Scale Devices. In *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVI, pages 159–170, New York, NY, USA, 2011. ACM.

[36] J. Ryoo, J. Jian, A. Athalye, S. R. Das, and M. Stanaćević. Design and Evaluation of 'BTTN': A Backscattering Tag-to-Tag Network. *IEEE Internet of Things Journal*, 2018.

[37] J. Ryoo, Y. Karimi, A. Athalye, M. Stanaćević, S. R. Das, and P. Djurić. BARNET: Towards Activity Recognition Using Passive Backscattering Tag-to-Tag Network. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pages 414–427. ACM, 2018.

[38] Saleae. Logic 8. `support.saleae.com/datasheets-and-specifications/datasheets`. Last accessed: Jun. 2018.

[39] H. H. R. Sherazi, L. A. Grieco, and G. Boggia. A Comprehensive Review on Energy Harvesting MAC Protocols in WSNs: Challenges and Tradeoffs. *Ad Hoc Networks*, 71:117–134, 2018.

[40] Y. Sun, O. Gurewitz, and D. B. Johnson. RI-MAC: A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, pages 1–14, New York, NY, USA, 2008. ACM.

[41] TI Inc. CC1101 Low-Power Sub-1GHz RF Transceiver. `www.ti.com/lit/gpn/cc1101`. Last accessed: Sep. 2018.

[42] TI Inc. INA225EVM. `www.ti.com/lit/ug/sbou140/sbou140.pdf`. Last accessed: Jun. 2018.

[43] TI Inc. MSP430FR5994 16 MHz Ultra-Low-Power MCU. `www.ti.com/lit/gpn/msp430fr5994`. Last accessed: Sep. 2018.

[44] L. Van Hoesel and P. Havinga. A Lightweight Medium Access Protocol (LMAC) for Wireless Sensor Networks. In *1st Int. Workshop on Networked Sensing Systems (INSS 2004)*, 2004.

[45] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1567–1576. IEEE, 2002.

[46] K. S. Yildirim, H. Aantjes, P. Pawelczak, and A. Y. Majid. On the Synchronization of Computational RFIDs. *IEEE Transactions on Mobile Computing*, pages 1–1, 2018.

[47] K. S. Yıldırım, R. Carli, and L. Schenato. Adaptive Proportional-Integral Clock Synchronization in Wireless Sensor Networks. *IEEE Transactions on Control Systems Technology*, 26(2):610–623, 2018.

[48] P. Zhang and D. Ganesan. Enabling Bit-by-Bit Backscatter Communication in Severe Energy Harvesting Environments. In *NSDI*, pages 345–357, 2014.

[49] P. Zhang, P. Hu, V. Pasikanti, and D. Ganesan. Ekhonet: High Speed Ultra Low-Power Backscatter for Next Generation Sensors. In *Proceedings of the 20th annual international conference on Mobile computing and networking*, pages 557–568. ACM, 2014.

[50] P. Zhang, M. Rostami, P. Hu, and D. Ganesan. Enabling Practical Backscatter Communication for On-Body Sensors. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 370–383. ACM, 2016.

[51] X. Zhou, A. Boukerche, and M. B. Younes. An Adaptive Traffic Energy-Efficient MAC Protocol for Mobile Delay-Tolerant Sensor Networks. In *Global Communications Conference (GLOBECOM), 2016 IEEE*, pages 1–6. IEEE, 2016.