Continuous Sensing on Intermittent Power

Anonymous Author(s)

ABSTRACT

The main obstacles to achieve truly ubiquitous sensing are (i) the limitations of battery technology - batteries are short-lived, hazardous, bulky, and costly - and (ii) the unpredictability of ambient power. The latter causes sensors to operate intermittently, violating the availability requirements of many real-world applications.

In this paper, we present the *Coalesced Intermittent Sensor* (CIS), an intermittently powered "sensor" that senses continuously! Our key observation being that, the power cycles of energy-harvesting battery-less devices do not show correlation even when they are drawing energy from the same source, running the same application, and in close proximity. We challenged our observation using different real hardware and energy sources and showed that this observation holds.

Another important finding is that a CIS designed for certain (minimal) energy conditions requires no explicit spreading of awake times due to embedded randomness in the powering subsystem. However, when the available energy exceeds the design point, nodes employing a sleep mode (to extend their availability) do wake up collectively at the next event. This synchronization leads to problems as multiple responses will be generated, and -worse- subsequent events will be missed as nodes will now recharge at the same time. To counter this unwanted behavior we designed an algorithm to estimate the number of active neighbors and respond proportionally to an event. We show that when intermittent nodes randomize their responses to events, in favorable energy conditions, the CIS reduces the duplicated captured events by 50% and increases the percentage of capturing entire bursts above 85%.

CCS CONCEPTS

• **Human-centered computing** → *Empirical studies in ubiquitous and mobile computing*.

KEYWORDS

Embedded systems, Energy harvesting, Ubiquitous computing, Intermittent Systems

ACM Reference Format:

Anonymous Author(s). 2019. Continuous Sensing on Intermittent Power. In *Proceedings of ACM Conference (Conference'20).* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Conference'20, April 2020, Sydney, Australia

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Batteries may compromise the viability of sensor nodes in various ways. Batteries are bulky, short-lived, hazardous, and expensive. To ameliorate the battery problem, researchers have been investigating different alternatives to extend lifetime and reduce costs and form factor. The reduction in power consumption of recent microcontrollers and the advances in energy-harvesting circuitry have enabled the emergence of battery-free energy-harvesting sensors. These sensors elide the constraints of batteries and extract power from ambient energy sources such as sunlight and RF emissions.

Ambient energy sources provide perpetual power. However, ambient power is usually too weak to directly power a sensor node [19]. Therefore, an energy-harvesting node first buffers the harvested energy until a usable amount has been accumulated; then it operates, for a short period of time, until the buffered energy has been exhausted [20]. Consequently, battery-less energy-harvesting sensors operate intermittently (Figure 1).

Intermittent power introduces a set of new challenges that are under ongoing investigation. For example, [3, 6, 20, 27, 28] studied the intermittent computation problem, which is concerned with the preservation of application progress and data consistency under frequent power failures; Hester et al. [12] investigated the timely operation challenge, which is concerned with data freshness after a power interrupt; and Yıldırım et al. [37] introduced event-driven execution for the intermittent domain, which deals with input and output operations under arbitrarily-timed power loss.

Despite these notable advances, intermittently-powered sensors suffer from a new fundamental shortcoming: *the intermittent availability of the system.* Being frequently off charging compromises the value of these devices. For example, a sensor that has a low probability (e.g., 10% [22]) to be available (on) when an event of interest occurs has no value. Overcoming the intermittent availability challenge without changing the size of the device or re-including batteries requires a novel approach that explores new design dimensions.

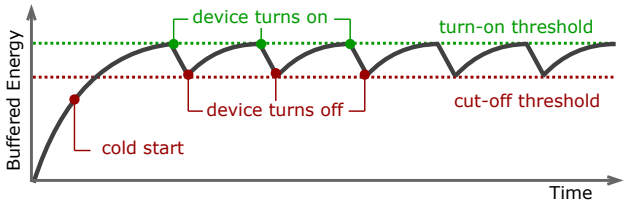


Figure 1: Harvested-energy profile. Ambient power is weak; therefore, it is usually buffered. The buffered energy is then consumed to operate the device. The operation period is often short as power consumption is much higher than the energy harvesting rate.

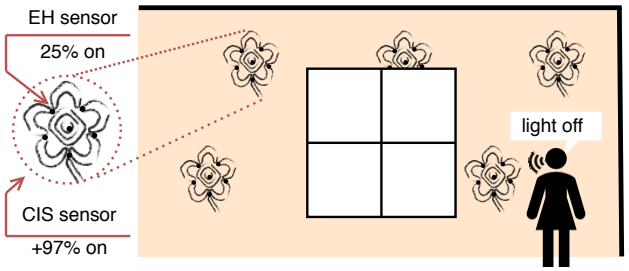


Figure 2: A Coalesced Intermittent Sensor (CIS) is a group of intermittently-powered nodes that sense continuously despite the intermittent power supply. CIS exploits the inherent randomization of energy harvesting systems, if available, and introduces artificial randomization, when needed, to preserve continuous sensing.

1.1 Vision and Application

Miniaturized sensors are less intrusive devices than bigger ones. Therefore, they can be embedded in locations that are not suited for the others (enabling new applications). Miniaturizing sensors, however, signifying their powering challenge. On one hand, batteries make these sensors *continuously* available -for sensing opportunities- but for a short period of time (even rechargeable batteries wear out after a few hundreds of charging cycles [1]). On the other hand, removing batteries and relying on ambient energy make them available for *extended period of time* but intermittently. Our vision is that by combining multiple battery-less energy-harvesting sensors we can create a new virtual sensor that operates permanently (no batteries) and reliably (continuously available): we call this sensor the *Coalesced Intermittent Sensor* (CIS).

Sensors with such characteristics would allow us to add a cheap and maintenance-free sensing layer to many objects, making them smart and interactive. For example, one can imagine developing smart wallpaper that users can interact with. Smart wallpaper with embedded microphones can enable direct in-building human-to-object communication (Figure 2). Such a permanently operating sensor can be deployed, for example, in kids' playgrounds to monitor their occupancy. These battery-less sensors can enable interactive and safe-to-dispose sports rugs (that count how many times a person has jumped on them) or play rugs for kids. In short, we would like to develop small sensors with permanent and continuous sensing capabilities.

1.2 Research Challenges

Many sensing applications require the sensor to be available when there is a change in the monitored environment. Energy-harvesting battery-less sensors can provide cheap and maintenance-free sensing, but the do not meet the availability requirements of many real-world applications.

C1-Approach continuous availability on intermittent power: An energy-harvesting battery-less sensor is frequently off, spending most of the time charging. One way to increase the system availability is by using multiple nodes. However, coordinating the nodes' awake times using communication may introduce prohibitive overhead as a scattering algorithm must be regularly executed, and messages for synchronizing nodes' clocks and reserving time slots need to be repeatedly exchanged. Thus, the challenge is *can we exploit some of the inherent characteristics of energy-harvesting battery-less sensors to distribute nodes' awake times without the need for communication?*

C2-Continuous sensing on intermittently powered sensors: Even when the collective availability of intermittent sensors approaches 100%, the emerging overall sensing behavior may still be intermittent. Event-trigger sensors sleep in low-power mode waiting for an event to wake them up. When ambient energy rises, the energy-harvesting rates of these sensors may equal (or approximate) their sleeping mode power consumption. Under such energy conditions, these sensors become available for an extended period of time. Therefore, when an external event arrives, nodes respond collectively, which exhausts their energy buffers, making them unavailable for the next set of events. This is, particularly, a significant problem when events arrive in bursts, like a command of a few words (e.g., light on). Thus, the challenge is, *how to prevent energy-harvesting battery-less sensors from synchronizing their power cycles on some of the incoming events?*

C3-Efficient sensing on intermittent sensors: One of the main factors that determine the intermittency pattern of an energy-harvesting battery-less sensor is the richness of ambient energy. For example, at mid-noon under direct sunlight, even a small solar panel can power a sensor node continuously. In such conditions (favorable energy conditions), using plenty of intermittent sensors would only result in duplicated work that leads to duplicated messages when the data is being communicated to a sink node: a continuously-powered node acts as a gateway for such sensors to communicate with other layers of the Internet of Things. These messages will collide as they will be generated at approximately the same time, and if some of them are received by the sink, then they waste energy as they carry the same information. Thus, the challenge is, *how to reduce the number of duplicated event detections?*

1.3 Contributions

In this paper, we tackle the paradox of continuous sensing on intermittently-powered sensors. We studied the inter-relationship between the power cycles of energy-harvesting battery-less devices, the emerging collective behavior, and the effect of the change in ambient energy on this behavior. In particular, this paper makes the following key contributions:

- We show how to approach continuous sensing using multiple intermittently-powered sensors. For that, we **modeled** the collective effective availability—the system availability that leads to successful sensing—of a group of intermittent sensors and **validated** our models using simulation and on real hardware against different ambient energy sources.
- We introduce a new type of virtual sensor, showing its capabilities and limitations. This **Coalesced Intermittent Sensor (CIS)** is the abstraction of a group of intermittently-powered sensors that achieves maximum statistical availability by exploiting (inherent) randomization to spread nodes' awake times uniformly.

- Contrary to common sense, we show how favorable energy conditions can deteriorate the performance of a CIS. We, therefore, equipped the CIS with **an new algorithm** that makes it ambient-energy aware. This algorithm enables the nodes to determine their own duty cycles (without requiring additional hardware), and the average number of alive nodes (without requiring communication). This information can effectively be used by the nodes to decide when to back off to avoid duplicated event detection and availability interruptions (implicit synchronization in favorable harvesting conditions).
- We prototype, evaluate, and demonstrate the feasibility of the Coalesced Intermittent Sensor concept in the form of a voice-control commands recognizer, the **Coalesced Intermittent Command Recognizer** (**CICR**). We chose to develop a command recognizer as voice is a natural way for the human to interact with small devices. Moreover, words allow us to easily experiment with individual event arrivals and events that arrive in bursts. However, the goal of this paper is *not* to present a novel word recognition technique. Instead, we adapts a classical word recognition algorithm to make it power-failure immune. Yet, our CICR prototype is the first intermittent command recognizer, shedding light on the potential of intermittent systems.

2 RELATED WORK

Recent advances in ultra-low-power microcontrollers along with the development of energy harvesters have enabled the creation of stand-alone battery-free sensors. These sensors operate intermittently because the power that they harvest is weak and volatile.

2.1 Energy-harvesting systems

Energy harvesters have the potential to power devices indefinitely as they collect energy from perpetual energy sources. Sunlight, vibration, and radio frequency (RF) waves are examples of such energy sources. The power harvested from these sources vary wildly, for example, RF harvestable power ranges from nW-scale when harvested from ambient signals to µW-scale when collected from a dedicated RF signal emitter, and solar power varies from tens of µW to tens of mW when it is harvested by a solar panel of a few cm^2 illumination surface [20, 29].

Many battery-less energy-harvesting platforms have been proposed. Some of them rely on dedicated external energy sources such as WISP -and its variants-, a general wireless sensing and identification platform [31, 38, 39]; WISPcam, an RF-powered camera [25] and, the battery-free cellphone [32]. Others, harvest from ambient sources such as the ambient backscatter tag [19], and the solar-powered tag [23]. Platforms that facilitate the development of battery-less energy-harvesting systems have also been proposed. For instance, Fliker [11], a prototyping platform for battery-less devices; EDB [5] an energy-interference-free debugger for intermittent devices; and Capybara [7], a re-configurable energy storage architecture for energy-harvesting devices.

However, *there is no energy-harvesting platform that considers the abstraction of many intermittent sensors (or nodes) and exploits the statistical energy harvesting differences between them to provide reliable sensing.*

2.2 Intermittent execution

Intermittent execution models enable applications to progress despite frequent power failures [4, 6, 10, 21, 35]. To this end, they decompose an application into several small pieces and save the state of the computation on the transitions between these code segments. Therefore, intermittent applications do not return to the beginning of the program (i.e., `main()`) after each power failure (in contrast to applications that assume continuous power). Instead, they resume execution from the last successfully saved progress state.

Mementos [28] proposed a volatile memory *checkpoint-based* approach to enable long-running applications on intermittently powered devices. DINO [27] enables safe non-volatile memory access despite power failures. Chain [6] minimizes the amount of data needed to be protected by introducing the concepts of *atomic tasks and data-channels*. Hibernus [2, 3] measures the voltage level in the energy buffer to reduce the number of checkpoints per power cycle. Ratchet [35] uses compiler analysis to eliminate the need for programmer intervention or hardware support. HarvOS [4] uses both compiler and hardware support to optimize checkpoint placement and energy consumption. Mayfly [12] enables time-aware intermittent computing. InK [37] introduces event-driven intermittent execution. *Our system adopts a power failure protection approach similar to that of DINO [27].*

2.3 Speech recognition

The speech recognition problem has been tackled from many angles and has experienced many breakthroughs. For example, the dynamic time warping (DTW) algorithm enables matching voice signals with different speed (or time) [36]. Approaches based on Hidden Markov Models showed much better performance than DTW-based ones [18]. Hence, they became the standard techniques for general-purpose speech recognition until artificial intelligent algorithms [13] outperform them.

From a recognition complexity standpoint, we can classify the speech into *spontaneous speech, continuous speech, connected word,* and *isolated word* [8]. The *continuous* and *spontaneous speech* are the closest to natural speech, but they are the most difficult to recognize because they need special methods to detect words boundaries [8]. This is less the case for the *connected word* type, where a minimum pause between the words is required. The type with the least complexity is the *isolated word*, as it requires a period of silence on both sides of a spoken word.

Speech recognition on resources—memory, computation power, and energy—limited platforms is challenging, to say the least. Therefore, *our command recognizer targets isolated-word type of speech.*

3 COALESCED INTERMITTENT SENSOR

The Coalesced Intermittent Sensor (CIS) is the abstraction of a group of energy-harvesting battery-less sensor nodes seeking to approximate the continuous sensing availability characteristic of a battery-powered sensor. The design of a CIS needs to consider four main aspects: (i) how the nodes' awake time is distributed;

(ii) the consequence of emulating continuous sensing availability by chaining multiple short on-times; (iii) the effect of the environment on the CIS's availability; and (iv) the spacial coverage of the event of interest, which determines the diameter of the CIS.

However, let us first characterize the power cycle of an energy-harvesting battery-less device. An energy-harvesting intermittent node frequently switches between off and on, charging energy and operating. We can characterize, from a time perspective, this charge-discharge (or power) cycle using the following notation, $(t_{\text{on}}, t_{\text{p}})$, where t_{on} is the node's uptime interval, and $t_{\text{p}} := t_{\text{on}} + t_{\text{off}}$, where t_{off} is the node's charging time interval.

3.1 Sensing

The ability of a CIS to sense depends on the availability of its intermittent nodes and on the characteristics of the event of interest.

3.1.1 Coalesced availability. The CIS's availability is the projection of its underlying intermittent nodes' on-times on the time axis. To determine the expected availability of a CIS, the strategy being employed to distribute its nodes' on-times must be first specified.

Explicit on-time division strategy. A CIS can build on top of the recent advancements in passive (light or RF) communication [19**?**] and ultra-low-power timers [12] to apply a time-division multiplexing strategy, minimizing on-times overlapping. For example, a node calculates its average on-time $\overline{t_{\text{on}}}$ and off-time $\overline{t_{\text{off}}}$ for a certain number of power cycles. Then, it encodes the information $(\overline{t_{\text{off}}}, \overline{t_{\text{on}}})$ in a message and broadcasts it at the beginning of its power cycle. When a node receives this message it will have full knowledge about the transmitting node's power cycle. It can then alter its power cycle, relative to the transmitting nodes cycle, by increasing (or decreasing) its power consumption to shorten (or lengthen) its on-time and subsequently shift its power cycle to a different time slot.

With such explicit on-times control strategy, a CIS of N nodes with on-time of $\overline{t_{\text{on}}}$ and off-time of $\overline{t_{\text{off}}}$ will have an availability $= \min\left(N \times \frac{\overline{t_{\text{on}}}}{\overline{t_{\text{p}}}}, 100\%\right)$. However, we expect such an approach to introduce significant overhead as a scattering algorithm (e.g., [9]) must be frequently executed, messages need to be exchanged, and clocks should be synchronized. Therefore, we propose a different on-times spreading strategy.

Implicit on-time division strategy. With no information being exchanged between intermittent nodes, the best CIS can do is to uniformly distribute its node's on-times and maintaining this distribution over time. The key observation to approach uniform distribution is to ensure that the lengths of the node's power cycles are randomized, avoiding nodes being in lockstep indefinitely.

Let us start by assuming that we have a CIS of two nodes with idealized power cycles and these nodes have the same initial conditions. The availability of this CIS equals t_{on} as the nodes are in perfect synchronization (the two nodes wake up and power down together). To extend the availability of this CIS, one of the node should shift its on-time away from the other. If one of the nodes sleeps for t units of time, then the on-time of this power cycle will be $t_{\text{on}} + \Delta t$. Consequently, the length of this power cycle will be

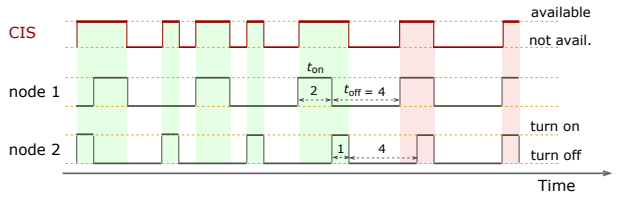


Figure 3: A Coalesced Intermittent Sensor's availability is the emerging collective on-time of its intermittent nodes' on-times. The difference between the power cycles leads to a constant relative shift between the nodes duty cycles. This, in turn, causes their on-times to be uniformly distributed on the overall power cycle. The red bars indicate a minimum CIS time span—CIS's nodes are overlapping—whereas the green bars show the maximum time span of the CIS.

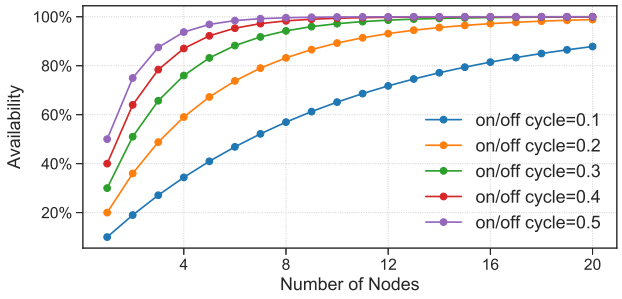
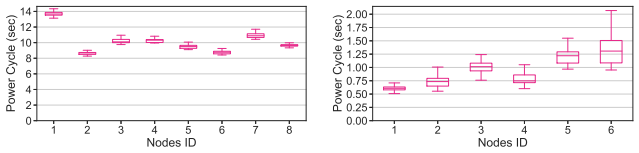


Figure 4: Coalesced Intermittent Sensor availability for a different number of nodes and different duty cycles. The nodes are uniformly distributed and the CIS on-time evolves, when adding new nodes, according to equation 1.

$t_{\text{p}} + \Delta t$, delaying the next awake time by Δt. If the node sleeps only once, then availability of the CIS will equal $\min(2 \times t_{\text{on}} | t_{\text{on}} + \Delta t)$

However, if the initial conditions are unknown, then shifting a node's on-time a constant number of times may cause the initially desynchronized nodes to become synchronized, collapsing the CIS's availability instead of extending it. Therefore, a safer option is to *constantly* shift the awake time of the node. In this case, the on-time will shift over the entire power cycle of the other node, spending $\frac{t_{\text{off}}}{t_{\text{p}}}$ and $\frac{t_{\text{on}}}{t_{\text{p}}}$ of the time overlapping with the other node's off-time and on-time, respectively. This behavior is illustrated in Figure 3, where node 1 and node 2 have power cycles of (2,6) and (1,5). Following the time axis from the left to the right, we can observe that the position of the on-time of node 2 is shifted by -1 unit of time relative to the on-time of node 1 after each power cycle of node 1. This implies that the on-times of the two nodes are $\frac{1}{3}$ of the time cluster together and $\frac{2}{3}$ of the time they are apart (from an external event standpoint, the on-times are uniformly distributed over the longest power cycle, as they have the same probability to be anywhere when the event arrives). To model the availability of a CIS of N nodes, we first model the nodes' on-times and power cycles. If we represent the on-time of a node with a random variable R_{n} and find



(a) Light (680μF) **(b) RF (47μF)**

Figure 5: Nodes' power cycles length for different ambient energy sources, and different energy buffer sizes.

its expected value $E(R_n)$ then we can approximate *any* CIS node's on-time with mean of the expected values of the nodes' on-times, i.e., $t_{on} = \frac{1}{N} \times \sum_{i=1}^{N} E(R_n)^i$ (intuitively, since we are assume CIS's nodes have the same energy buffer, then their expected on-times should approach the same value). Using a similar analogy, we can define the mean of the expected values of the power cycles lengths as $t_p = \frac{1}{N} \times \sum_{i=1}^{N} E(R_p)^i$. Now, we can model the availability of a CIS of N nodes as:

$$A_v(N) = A_v(N-1) + (1 - A_v(N-1)) \times \frac{t_{on}}{t_p}, \quad (1)$$

for the initial case where $N = 1$ we define $A_v(0) := 0$. Figure 4 shows the availability of CIS when $N \in \{1, 2, .., 20\}$ and nodes' duty cycles $\frac{t_{on}}{t_p} \in \{10\%, 20\%, .., 50\%\}$. We can conclude from the above discussion that to approach uniform distribution of nodes' on-times, the lengths of the power cycles need to be randomized [1].

The power cycles of energy-harvesting battery-less devices are inherently randomized and different because the power source (ambient energy) is volatile and the harvesters are not perfect devices (notice that, even battery-powered wireless sensor nodes require a synchronization protocol to correct for the drift in their local clocks). Our own measurements using different energy-harvesting devices and different energy sources, i.e., solar and RF, also confirm that the power cycles of intermittent nodes are different and randomized (Figure 5). Therefore, we expect their on-times to be uniformly distributed (we will challenge our expectation in Section 5).

3.1.2 Events classification. The availability of a CIS is not a single stretched interval: it is a chain of short intervals. Therefore, it is important to classify from a CIS perspective which types of events the CIS is best suited for.

- *Short events*: are events that can be captured using single intermittent node. For example, a spoken word can be seen as a short event if the energy needed to record it is less than what the energy buffer, i.e., the capacitor, can store.
- *Long events*: are events that need more energy to be completely captured than what the energy buffer can store. Long events can be subdivided into three categories:
  - *Simple*: is a long event that can be captured using single intermittent node—capturing part of it is sufficient to obtain all the information of interest—such as the sound produced by the friction between two moving parts of an engine (*Why this is not a short event? keep reading*).
  - *Burst*: is a group of short events that requires multiple intermittent nodes to be captured such as a command of a few words (e.g., room temperature up).
  - *Complex*: is a long event that must be fully captured to be recognized. For example, sampling a gyroscope attached to a moving device (e.g., a toothbrush).

[1]Note that, having power cycles of lengths that are multiples of each other is a very unlikely as nodes' energy buffers are assumed to be of the same size.

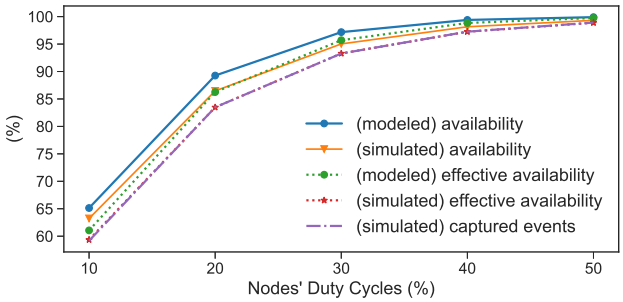


Figure 6: Simulating the availability, the effective availability, and successfully captured events of a CIS of 10 nodes with a node duty cycle $\in \{10\%, 20\%, ..., 50\%\}$.

Based on the above classification, we can argue that designing a CIS for long events is not like designing it for capturing short ones. For example, while capturing a short event may require continuous CIS availability, capturing a long simple event that is longer than the power cycle t_p does not require extending the availability of a single intermittent node. Furthermore, capturing a long complex event may require data fusion and processing that require the CISs' nodes to communicate the raw data to a more powerful node, which may lead to significant overhead. However, this paper focuses on short and long bust events as they cover a wide range of applications (e.g., voice-controlled human-object interface).

3.1.3 Effective Availability. Approaching continuous availability does not mean that a CIS can successfully capture all events. It can happen that an event is being only partially captured by one or more nodes, which may lead to unsuccessful event detection. Therefore, it is important to specify the effective availability of a CIS that leads to a successful event capturing (which we assume leads to successful sensing).

polling-based Sensing. Let us assume that we have a CIS of a single intermittent node monitoring a short event of length t_e. For capturing the entire event, the event has to arrive within the interval, $t_{on} - t_e$, which we call, the effective on-time of an intermittent node. Therefore, the effective availability of a CIS of N nodes is the joined effective on-times of the underlying intermittent nodes, which can be modeled as,

$$A_v(N) = A_v(N-1) + (1 - A_v(N-1)) \times \frac{t_{on} - t_e}{t_p}, \quad (2)$$

Event-driven Sensing. An intermittent sensor has a limited energy budget per power cycle. When it is tasked with a polling-based sensing activity, its energy consumption, generally, switches between two levels: zero when charging and maximum when sensing. However, in event-based sensing, a node puts its microcontroller

into low-power mode and waits (or listens) for an external event to wake up the microcontroller. For example, in our prototype, a voice-controlled command recognizer, we exploit the microphone's wake-on-sound feature to send an interrupt to the microcontroller, which will then start recording the sound samples from the microphone. This wake-on-event style of operation is important as the minimal energy consumption during sleep significantly prolongs the period during which an event can be handled (for example, our prototype consumes 7 times less energy during sleep compared to being active). To model the effective CIS availability when it is tasked with event-based sensing the change in energy consumption between the sleep and active mode must be taken into account. Since the event itself times when the node changes its energy consumption, we can model the effective availability as,

$$A_v(N) = A_v(N-1) + (1 - A_v(N-1)) \times \frac{t_s - (t_e \times \frac{p_a}{p_s})}{t_{sp}}, \quad (3)$$

where t_s is the expected sleep time of the CIS's nodes, $t_{sp} \coloneqq t_s + t_{off}$, and p_a and p_s are the power consumption at active and sleeping mode, respectively. Notice that, there is a subtle point about equation 3 as when an event arrives the node wakes up, consuming more energy. Therefore, its uptime shrinks. We, for simplicity, modeled this effect by extending the event time with the same factor. This is sufficient to say if that the event will be fully captured or not (effective availability).

3.1.4 Simulation. To do a first sanity check on our models, we simulated 10^5 power cycles of a CIS of 10 nodes (Figure 6). The duty cycles of the nodes range from 10% to 50%, while the event length is fixed at 3% of the power cycle length, t_p. The on-times and events arrivals were uniformly distributed over the power cycles. The results clearly confirm our models and support our argument about the distinction between CIS's availability and effective availability. The importance of this distinction is a function of the value $\frac{t_e}{t_{on}}$; observe the difference between availability and effective availability when nodes' duty cycle is 10% (large effect) and 50% (negligible effect).

3.2 Environment

Ambient energy controls the availability of a CIS's nodes. Consequently, it also controls their collective response to external events. When it rises, it extends nodes' on-times that may lead node's power cycles to be synchronized on the arrival of some external events, compromising the CIS's overall availability. To overcome this problem the CIS's nodes must be power-state aware and able to estimate the number of active nodes in the CIS.

3.2.1 Power States. A CIS can experience a wide range of ambient power intensities. For example, a solar-powered CIS may harvest no energy at night, modest energy from artificial light, and abundant energy from direct sunlight. Generally, we can identify four different CIS powering states:

- *Targeted power state*—These are the powering conditions that a CIS is designed for. In these conditions, the CIS should work intermittently and have sufficiently randomized power cycles to uniformly distribute its intermittent nodes on-times and meet the desired availability (Figure 4). In general, the targeted powering conditions should be near worst energy harvesting conditions to ensure that the system is properly functioning for the majority of the time.
- *Under-targeted power state*—Ultimately, the ambient energy is an uncontrollable power source, and it is not hard to imagine scenarios where a CIS will be under-powered or even comes to complete and long power down (for example, a solar CIS will come to a perpetual power down in darkness). In general, for under-targeted energy conditions, the CIS behavior can be considered as undefined.
- *Hibernating power state*—In event-driven sensing, nodes sleep in low-power mode waiting for an event to wake them up. This mode extends CIS's nodes' on-times and makes them overlap significantly. Moreover, the on-times overlap even more, when ambient energy level rises (favorable energy conditions). If an event arrives in such conditions, it will wake up many nodes, causing them to consume their buffered energy much faster. Some of these nodes will power down before capturing the entire event, while others survive. If we assume that nodes have the same energy buffers, then the nodes that died before finishing the event should have been sleeping for a *longer* time than the nodes the capture the entire event (because they slept for a long time, they could not finish capturing the event). However, nodes that captured the entire event have a *shorter* overall uptime than the nodes the have died while capturing the event because they stayed *longer* in the active mode (because they stayed longer in active mode, they consume their energy budget faster). This will lead the nodes' power cycles to tend to synchronized after the event. Let us analyze the example presented in Figure 7 to see exactly why nodes' power cycles tend to synchronized after an event if some or all of the nodes die while capturing an event. The figure shows the power traces of two nodes. It indicates that the nodes consume 7 times

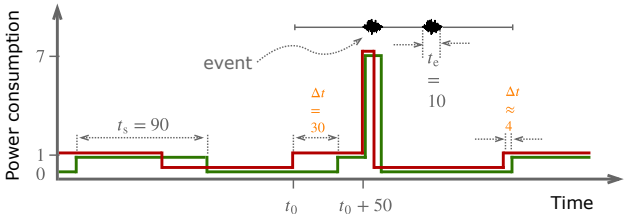


Figure 7: Capturing events may lead to power cycles synchronization of nodes that were in low-power mode. In particular, if some of the nodes power down while capturing the event, then this implies that these nodes slept (prior to the event) longer than the nodes that capture the event. This means that the nodes that have died spent their energy slower (they stayed longer in sleep mode); therefore, their overall uptime is longer than the uptime of the node the captured the event. In other words, the nodes that the woke up earlier have stated longer on, and therefore, the next power cycles are more synchronize, see the Δt before and after the event.

more power at active mode than that of the sleep mode (our prototype has a similar power consumption ratio, Table 3). Further, shows that a node sustains the low-power mode for 90 units of time, $t_s = 90$; therefore, the maximum buffered energy can be calculated as

$$E_{\text{buf}} = t_s \times p_s$$

, where p_s is a node power consumption at sleep mode. If we focus on the power cycle with the first event, then we see that node R powers up at t_0, and it remains in low-power mode for 50 units of time, whereas node G spends only 20 units of time before the event arrives. Now, we can calculate when these two nodes will power down and compare the difference, Δt, before and after the event. A node will turn off when the buffered energy is depleted. This can be expressed as follows,

$$E_{\text{buf}} = t_{\text{se}} \times p_s + t_{\text{on}} \times \delta p_s,$$

where t_{se} is the sleep time of the power cycle that an event arrives in, t_{on} is a node's on-time and $\delta = \frac{p_a}{p_s}$ where, p_a is the power consumption at the active mode. p_s can be eliminated and t_{se} and δ are given; thus to find when the nodes will power down when need to find t_{on} for both nodes. By substituting the given values we find t_{on} to be 10 and 5.7 units of time for the G and R node, respectively. Therefore, Δt becomes 4.3 while it was 30 before the event.
When the events arrive in burst this becomes a significant problem, as a CIS will capture multiple copies of the first event, while missing the subsequent ones (Figure 7).

- *Continuous power state*—Under direct mid-noon sun a tiny solar panel may provide sufficient power to run a sensor node continuously. In such conditions, a CIS's node will be available and able to sense continuously. Therefore, the job of a single node will be repeated N times, and instead of sending a single message to a sink—to push the data to the Internet—N identical messages will be sent. These messages will collide as they are sent at about the same time, causing the information to be lost; if they arrive, however, they - except the first one- will waste energy of the sink as they carry the same information.

The inefficiencies highlighted in the hibernating and continuous power states can be mitigated by enforcing randomization on the response of intermittent nodes : when a node is woken up by an external event it responds to that event with a certain probability. However, if the randomized response is enforced all the time, then the CIS will have a lower probability of catching events during the targeted energy conditions state. Therefore, the CIS has to distinguish between the targeted and above-targeted energy conditions and randomize its response only during the hibernating and continuous power states.

Furthermore, responding with a constant probability during the above-targeted energy conditions is inefficient, as the number of active nodes is a function of the total number of intermittent nodes and the power intensity at that time. Therefore, efficient randomization requires intermittent nodes to estimate the number of active nodes and respond proportionally. Our proposed algorithm

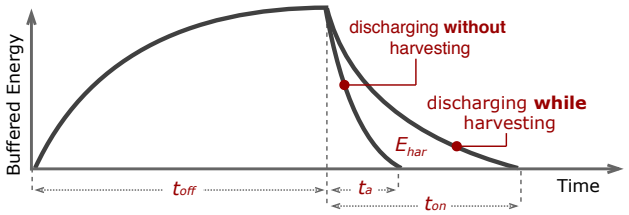


Figure 8: The difference in the time of discharging the energy buffer—a node's on-time—when an energy-harvesting device is allowed to charge while operating, and when it is not allowed.

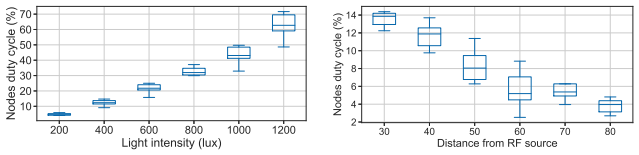
Algorithm 1 off-time estimation

```
1:  R_cntr++                                   ▷ reboot counter
2:  E_buf                                      ▷ Size of energy buffer
3:  t_a                                        ▷ time of discharging E_buf at load a, no harvesting
4:  X_cy                                       ▷ time every X power cycles
    ▷ Code executed on each X power cycles
5:  if (R_cntr == X_cy) then
6:      f_LOAD(a)                              ▷ set node load to a
7:      t_on ← TIME()                          ▷ measure time until power down
8:  end if
    ▷ Code executed on each X + 1 power cycles
9:  if (R_cntr == X_cy + 1) then
10:     Δt = t_on − t_a                        ▷ time difference due to charging
11:     E_har ← E_buf × t_a/Δt                 ▷ harvested energy
12:     P_in ← E_har ÷ t_on                    ▷ incoming power
13:     t_off ← E_buf ÷ P_in
14:     R_cntr = 0
15: end if
```

for estimating the number of active nodes depends on the nodes ability of measuring their on-times and off-times.

3.2.2 Intermittent Timing. Timing is a key building block of sensing systems. It is, however, missing on intermittent nodes unless an additional dedicated (RC-based) timer is included [12]. Here we propose an alternative that does not require additional hardware. This alternative does not only enable time estimation but also ambient energy richness, which is very important for estimating the number of a node's active neighbors. But, *how a node can time its own on/off cycle?*

Intermittent nodes fail abruptly; therefore, a persistent timer is needed to measure node's on-time. A simple way to emulate persistent timer is by using a persistent counter, or sampling the volatile built-in timers of the microcontroller and save the obtained values in the non-volatile memory. To estimate the off-time, t_{off} in Figure 8, a node needs to determine the incoming power (harvesting rate). The average harvesting rate can be induced from the on-time as follows. The node measures its on-time while harvesting, see t_{on} in Figure 8, and compares it to the time required to drain the energy buffer *without* charging, see t_a in Figure 8. The additional on-time, Δt, is the result of the energy accumulated while executing. If t_{on} and t_a are measured on the same load—thus, they have the same power consumption—then the amount of the energy harvested



(a) Light (b) RF

Figure 9: The average duty cycles of 8 solar-powered and 6 RF-powered intermittent nodes for different ambient energy sources and energy intensities. In general, the average duty cycle of a node is a good indicator of the average duty cycle of the other CIS's nodes.

Table 1: Measuring intermittent nodes overlapping of a CIS of 8 intermittent nodes for different light intensities.

light (lux)	on/off cycle (%)	N_{active}	std
300	8	1.01	0.85
500	17	1.63	0.98
800	31	2.88	1.50
1200	62	5.05	1.08

while the device is on can be calculated as in Line 11, Algorithm 1. And, the average input power can be found as in Line 12 that, in turn, enables the node to estimate its own t_{off} (Line 13). Since calculating the off-time requires constant load, the sensor cannot run arbitrary code during time measurement. Therefore, the sensor needs to sacrifice a certain percentage of its power cycles for measuring time (Line 1-7). Once the on-time and off-time are found the node's power cycle for load a is determined.

Notice that, when the harvested power is very low the accuracy of inferring the charging time from the discharging degrades. However, for the Coalesced Intermittent Sensor this is not a serious problem as the intermittent nodes need to randomize their response to events only in favorable energy conditions.

3.2.3 Alive nodes estimation. To estimate the number of active nodes, a CIS's node needs to determine the following information: (i) the total number of nodes in its CIS, which is a typically constant value that can be loaded to the device memory; (ii) the on-times distribution, which is uniform in our case; and (iii) its own average $\overline{t_{\text{on}}}$ and $\overline{t_{\text{off}}}$.

Since, we assume that a CIS's nodes have the same energy buffers and are in the vicinity of each other (thus, they are exposed to the same energy conditions) then their duty cycles should approach the same value. Figure 9 shows the average duty cycles of the nodes of a solar- and RF-powered CISs. In general, we can conclude that a node's average duty cycle is a good estimator of other CIS's nodes' duty cycles. Now, a node can estimate the maximum time span, t_{max}, of its CIS, which is the total duration of the nodes' on-times when they are aligned next to each other, as follows

$$t_{\text{max}} = N \times \overline{t_{\text{on}}}. \tag{4}$$

Then, from equation 1, the node calculates the CIS availability, $A_{\text{v}}(N)$. As we argued in Section 3.1.1, nodes on-times are uniformly distributed; therefore, the overlapping on-time is also uniformly

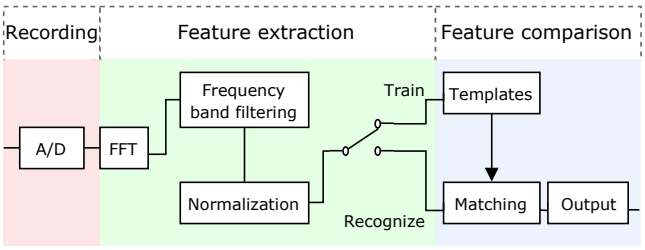


Figure 10: Coalesced Intermittent Command Recognizer: an instant of a Coalesced Intermittent Sensor. CICR features a power failure immune word recognition algorithm. First a word is recorded. Then, its spectral features are extracted. The resulting features vector is compared against previously-stored words' templates for recognition. The comparison is done using a liner distance matching algorithm

distributed. As such, a node can calculate the average number of active intermittent nodes, N_{active}, using the following formula,

$$N_{\text{active}} = \frac{t_{\text{max}}}{\overline{t_{\text{p}}} \times A_{\text{v}}(N)}. \tag{5}$$

3.2.4 Response randomization factor. Once a node has estimated the number of active neighbors, N_{active}, it can use the following formula to determine the response probability,

$$P_{\text{resp}} = \begin{cases} \frac{N_{\text{resp}}}{N_{\text{active}}} & \text{if} \frac{N_{\text{resp}}}{N_{\text{active}}} < 1 \\ 1 & \text{therwise,} \end{cases} \tag{6}$$

where N_{resp} is a system parameter that reflects the desired redundancy factor required by an application.

Table 1 shows the average number of active nodes of an 8-nodes CIS for different light intensities. These measurements provide a sanity check on equation 5. For example, at 1200 lux an individual node of our CIS has a duty cycle of $\approx 62\%$, i.e., it is on average $0.62\, t_{\text{p}}$ operating. If we multiply that by the number of nodes (equation 4) we get about $5\, t_{\text{p}}$. Figure 4 indicates that a CIS with eight nodes of duty cycles above 50% has near 100% availability. From equation 5, we find that the expected number of clustered nodes is 5 confirmed by the measurements presented in Table 1.

4 PROTOTYPE: COALESCED INTERMITTENT COMMAND RECOGNIZER

The coalesced intermittent command recognizer (CICR) is a prototype of the Coalesced Intermittent Sensor. The CICR consists of eight battery-less intermittent nodes. Each node is capable of performing isolated word recognition.

4.1 Hardware

A CICR node consists of thee main parts: a microphone, a microcontroller, and a harvester. MSP430RF5994 [34], an ultra-low-power microcontroller, is used for data acquisition and processing. This microcontroller has a 16-bit RISC processor running on 1 MHz, 8KB of SRAM (volatile), 256KB of FRAM (non-volatile), and a 12-bit analog to digital converter (ADC). It also features a Low Energy

Table 2: Testing set

on	off	stop	clear	load
go	pause	resume	edit	cancel

Table 3: Power usage

State	**Current** (μA)	**time** (ms)
Sleeping	64 ±20	—
Recording	423 ±20	285
Processing	282 ±20	600

Accelerator (LEA), which offloads the main CPU for specific operations, such as FFT. For recording we use the PMM-3738-VM1010-R piezoelectric MEMS microphone, which features Wake on Sound and ZeroPower listening technologies [26], allowing both the microcontroller and the microphone to sleep in a low-power mode until a sound wave is detected. The microcontroller and microphone are powered by a BQ25570 solar power harvester [33] connected to an IXYS SLMD121H04L solar cell [16] and a super-capacitor of 470 μF. For debugging we used the Saleae logic analyzer [30].

4.2 Software

The CICR runs power interrupts immune command recognizer. The recognizer is capable of recognizing isolated-word type of speech. The main parts of the recognizer are illustrated in Figure 10 and explained below:

Data acquisition. The *Wake-on-Sound* feature of the microphone triggers the data acquisition process once the energy level in the sound signal crosses a certain level. The ADC, then, samples the output of the microphone at 8 kHz. This sampling rate is sufficient to cover most of the frequency range of the human voice. The recording length was set to 285 ms, which suffices to get all the acoustic features needed to recognize the words.

Feature Extraction. For word recognition, we adopted the method presented in [14]. Here, we briefly describe the algorithm for convenience. The CICR starts by dividing the signal into frames of 256 samples (≈ 33 milliseconds). Then, it computes a 256-point Fast Fourier Transform for each frame. The resulting feature vectors are normalized (by computing the binary logarithm of each entry of that vector) to reduce detection errors that result from differences in the amplitude of the speech input. This feature vectors are the basis for the word-identification process.

Feature Matching. Feature matching is achieved by computing the the squared Euclidean distance between the normalized feature vectors of the recorded word and the feature vectors of the words stored during the training phase (templates, see Table 2). Once the recorded word has been compared to all template words, the template with the smallest distance to the recorded word is considered the correct word. However, if the smallest distance is bigger than a confidence threshold, then the CICR will return "undefined word".

We have experimented with two feature matching algorithms: the Linear Distance Matching (LDM) and Dynamic Time Warping (DTW) algorithm. While LDM compares the feature vectors of two words successively, DTW looks for the minimum distance between the two vectors. In our implementation, the DTW was about 10 times slower than LDM, whereas the detection accuracy was comparable; therefore, we default our implementation to LDM.

Power Failure Protection. In order to preserve the progress state and to protect CICR data against randomly timed power failures, we split the recognition program into 19 atomic regions. We ensured that each of these regions requires less energy than what the energy buffer can provide with a single charge. The program state is checkpointed in the non-volatile memory (FRAM) on the transition between these regions. This prevents the program from falling back to its starting point (`main()`) after each power failure. Data in the non-volatile memory with Write-After-Read dependency is double-buffered to ensure data integrity when the power supply is interrupted.

Code profiling. The entire command recognition software was written in C. The total program consists of 973 lines of code, excluding the FFT function, which is imported from the Texas Instrument DSP library. The memory footprint on the microcontroller is 20,064 bytes of FRAM and 1,134 bytes of SRAM.

The power usage of a node differs according to its activity. When a node is waiting for a voice event, it is in low-power mode. Recording a voice event activates the microphone, ADC and microcontroller (maximum power consumption). Processing the recorded data requires only the microcontroller to be on. Table 3 lists a node's power consumption for each of these states (sleeping, recording, and processing), as measured with a Monsoon power monitor [24].

5 EVALUATION

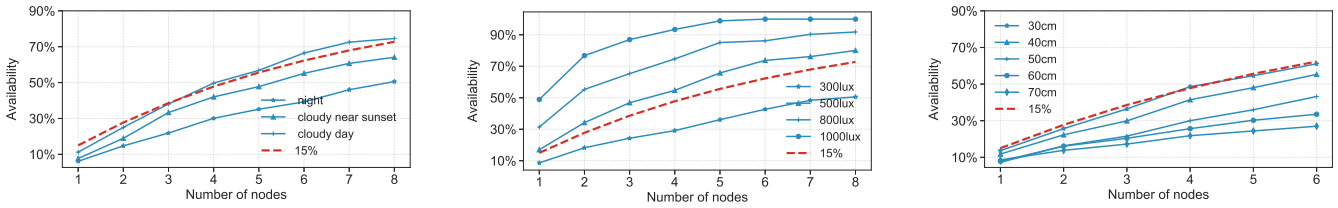
To evaluate the performance (availability) of the Coalesced Intermittent Sensor, we conducted several experiments in different energy conditions and with different event arrivals patterns.

5.1 Availability

Irrespective of the energy source (RF or light) we showed in Figure 5 that the power cycles of a CIS's nodes are different, which leads to a uniform distribution of their on-times, as we argued in Section 3.1.1. We captured the expected joined availability of these nodes in Model 1. Here, we validate model by comparing the modeled availability of a CIS against data captured with different hardware (solar- and RF-powered nodes) in different scenarios.

Figure 11 shows the availability of three CISs when they are powered by sunlight, artificial light, and RF and for a different number of intermittent nodes. The results clearly confirm our expectation: when the power cycles are slightly different, the on-times are uniformly distributed. The results also validate our model; the dashed lines represent the modeled availability when the nodes duty cycle is 15%.

5.1.1 Availability on a Fine Scale. Since the nodes' on-times are in a constant shift relative to each other (Section 3.1.1), the collective availability of the CIS fluctuates when it is observed in a short time window. Figure 12 captures CIS availability on a time window of 5 seconds for thee different ambient energy conditions. In these experiments, the average power cycles of the CIS's nodes are (3,18), (3.9,12.3), and (4.3,11.5) seconds when ambient light intensity are



(a) The CIS is powered by uncontrollable light sources—artificial light (night) and sunlight (day).

(b) The CIS is powered by a controllable array of LEDs.

(c) The CIS is powered by an RF reader [15] located 30-70 cm away from the RF tags (WISPs [31]).

Figure 11: The Coalesced Intermittent Sensor's availability for differed numbers of intermittent nodes and energy sources. The modeled availability—represented by the dashed lines when the average node duty cycle is 15%—approximates the measured availability with high accuracy.

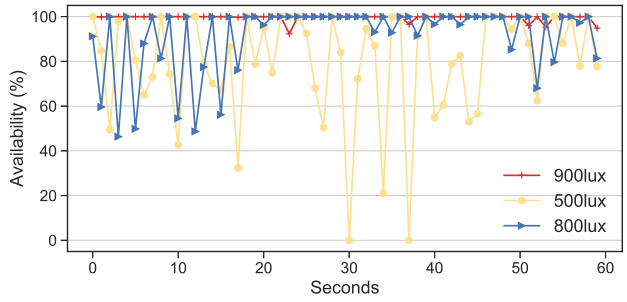


Figure 12: CIS availability observed with a 5 seconds time window.

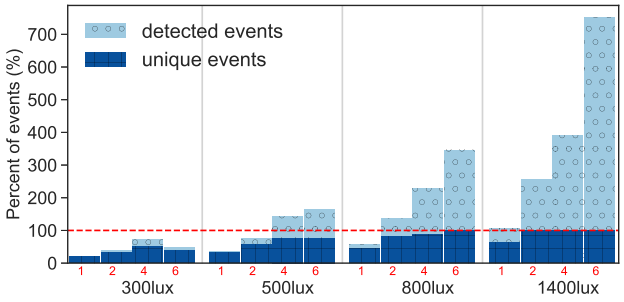


Figure 13: Duplicate and unique events captured by a coalesced intermittent command recognizer of eight solar-powered nodes. In general, the number of captured events increases in two case: when light intensity rises and when inter-event arrival time increases. Red numbers indicate events arrival interval in seconds.

500, 800, and 900 lux, respectively. If we focus on the line graphs associated with 500 and 800 lux and compare the system availability within the interval [20, 50] seconds and the rest, we can observe that the CIS gradually alternates between low and high collective availability; nodes' on-times gradually transition from maximum to minimum separation and vice versa (Section 3.1.1). Notice that, when ambient light intensity was 800 lux the CIS collective availability transited from low to high to low, while this pattern happened to be reversed when light intensity was 500 lux. For the 900 lux the 8-node CIS achieved near-continuous 100% availability.

5.2 Sensing

5.2.1 Experiment setup. After validating our observation on different energy sources, we designed a testbed with controllable light intensity for clarity and reproducibility. To this end, we blocked uncontrollable light sources with a box of $60 \times 40 \times 40$ cm. On the box ceiling, we attached a light strip of 2.5 m with 150 LEDs that can produce 15 different light intensities. On the bottom a Coalesced Intermittent Command Recognizer of 8 intermittent nodes is placed (see Section 4.1 for hardware description).

The events in our experiments are spoken words (Table 2). Short events (see events classification in Section 3.1.2) are represented with individual words, while burst events are represented with phrases of a few words. We recorded different patterns of inter-event and inter-bust arriving time. We used a Bluetooth speaker [17] to replay a certain recording. The data were collected using a logic analyzer [30] and processed on a laptop running Ubuntu 16.04 LTS.

5.2.2 Events detection rate. Here we experiment with the behavior of a CIS when events arrive individually or in bursts *without* enabling randomized response in favorable energy conditions.

Individual events. Figure 13 shows the percentages of capturing duplicate and unique events when light intensity varies from 300 lux to 1400 lux and the inter-event arrival time ranges from 1 sec to 6 sec. For each experimental trial 20 words were played, resulting in a total of 240 playbacks.

Figure 13 clearly shows a positive correlation between light intensity and the number of detected events. In particular, the number of duplicate detections rises dramatically when light intensity increases, *demonstrating the overpowering problem* (Section 3.2.1). Moreover, increasing the inter-event arrival time also surges the number of duplicated events. The reason for this phenomenon is that when the time between events increases, the intermittent nodes get the chance to sleep longer in low-power mode, consuming less energy. Therefore, nodes' on-times expand, reducing their inherent

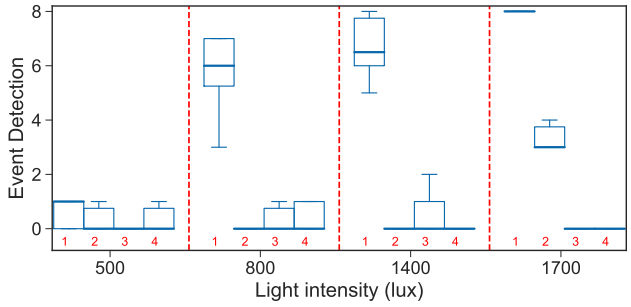


Figure 14: When capturing a burst of 4 events without randomizing the response, the majority of the nodes react to the first event in the burst and power down shortly after, missing the rest of the burst. Red numbers indicate events index in a burst.

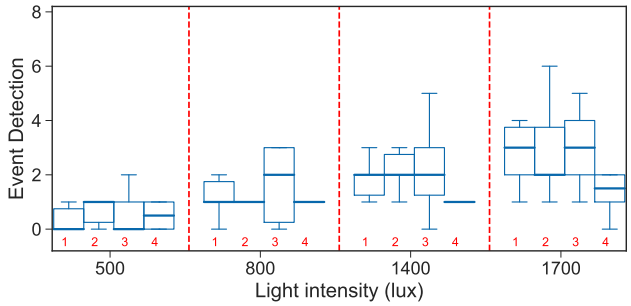


Figure 15: Response randomization enables a CIS to capture the entire burst of events with high capturing rates. It also reduces the number of duplicated events. Red numbers indicate events index in a burst.

randomization, which leads them to be in *hibernating power state* (Section 3.2.1).

Bursty events. Figure 14 shows the capturing behavior of a CIS when the events arrive in bursts. A burst of four events with one second between the individual events was fired every 20 seconds. Each burst was repeated 10 times and under four different light intensities. The nodes sleep in a low-power mode when they finish processing an event, waiting for the next one.

In general, we observe that in favorable energy conditions (above 500 lux) intermittent nodes react to the first event of a burst and power down shortly after, missing the rest of the burst. These results confirm our argument about the side effect of the *hibernating power state* of a CIS (Section 3.2.1). These results also demonstrate that the hibernating power problem happens on a wide range of power intensities, showing its significance. Next, we will show how randomizing the response can mitigate the problems generated when ambient energy exceeds the design point.

5.2.3 Events detection rate with randomization. Here, we examine the effect of enabling artificial randomization on the CIS's response.

(lux,second)	(800,6)	(1400,4)	(1400,6)
randomization	205/432	236/675	223/493
no randomization	240/831	240/938	240/1802

Table 4: These results are presented in the following format *unique/total* detected events. A CIS's node responds with a probability of 65% in the first two scenarios,(800,6) and (1400,4), and 30% for the last one. Randomizing the response reduces the number of duplicated events by 50% while losing only 7% of the unique events.

Individual events. Table 4 compares the number of detected events when the CICR's response is randomized and not randomized. When randomization is enabled, nodes respond to events with a probability of 65% for the scenario of (800 lux, 6 seconds) and (1400 lux, 4 seconds), and for the highest energy level and the longest inter-event arrival time the responding probability was set to 30%.

Table 4 shows that randomizing the response reduces duplicated events by an average of ≈50%, while only marginally lowers the number of the uniquely detected events (7% on average).

Bursty events. To enable a CIS to capture events arrive in bursts, the response probability for each events in a burst should be different. The CIS should respond with a minimum probability to the first event in a burst and gradually increase the response probability for the subsequent events in the burst (we assume that between the bursts the CIS resumes to its expected collective availability). This gradual increment to the responding probability is motivated by the observation that when a node captures an event it becomes unavailable for the subsequent ones in the burst. In this experiment, the nodes reacted with a probability of 40%, 50%, 70% and, 100% on the first, second, third and fourth event, respectively. Since the event distribution is known these probabilities were fixed during the development stage.

Figure 15 shows how randomizing the CIS response spreads the nodes' awake times–as compared to Figure 14–and enables the CIS to capture the entire burst with a high probability, i.e., above 85%. Additionally, we also observe a positive impact of randomizing the response when the system is under-powered (500 lux).

6 CONCLUSION AND FUTURE WORK

Energy-harvesting battery-less sensors can operate very long because their power source is unlimited. However, ambient power is weak and volatile; therefore, these sensors operate intermittently. The intermittent availability compromises their value as they have a high probability of missing events. This paper addresses the *availability* problem of intermittent sensors. It presents the *Coalesced Intermittent Sensor* (CIS), which is the abstraction of a group of intermittently-powered sensors. A CIS is able to approach continuous sensing by taking advantage of the embedded randomization in the powering subsystem of intermittent sensors. The resulting differences in the sensor nodes' power cycles make the nodes' on-times uniformly distrusted. Therefore, the number of a CIS nodes can be seen as a design parameter to achieve a targeted collective

availability. We have modeled the availability of a CIS and its effective availability: the availability that leads to successful event capturing. Further, we showed the accuracy of these models by validating them against data collected on real hardware and with different ambient energy sources (i.e., sunlight, artificial light, and RF). Furthermore, we showed how the variation in nodes' power consumption and harvesting rate and the arrival of external events can compromise the CIS's availability (nodes employing sleeping mode to increase the chance of successfully capturing an event, synchronize their power cycles on the first incoming event, in a burst, and miss the subsequent ones. The probability of this unwanted synchronization increases when ambient energy rises beyond the design point. To counter this unwanted behavior, we designed an algorithm to estimate the number of active neighbors and respond proportionally to an event. We developed a prototype of the CIS, an 8-nodes Coalesced Intermittent Command Recognizer (CICR). Using this prototype, we showed that the Coalesced Intermittent Sensor is able to distribute bursts of events on its nodes "evenly" and capture the entire burst with above 85% detection accuracy.

Intermittent sensors will partially capture events. Classical algorithms for recognizing and classifying these events might face difficulties dealing with partially captured data. Thus, follow-up work can investigate *how much machine learning algorithms can improve the sensing quality of intermittent sensing?*

REFERENCES

[1] Jayam Prabhakar Aditya and Mehdi Ferdowsi. 2008. Comparison of NiMH and Li-ion batteries in automotive applications. In *2008 IEEE Vehicle Power and Propulsion Conference*. IEEE, 1–6.

[2] Domenico Balsamo, Alex S Weddell, Anup Das, Alberto Rodriguez Arreola, Davide Brunelli, Bashir M Al-Hashimi, Geoff V Merrett, and Luca Benini. 2016. 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 (2016), 1968–1980.

[3] Domenico Balsamo, Alex S Weddell, Geoff V Merrett, Bashir M Al-Hashimi, Davide Brunelli, and Luca Benini. 2014. Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems. *IEEE Embedded Systems Letters* 7, 1 (2014), 15–18.

[4] Naveed Anwar Bhatti and Luca Mottola. 2017. HarvOS: Efficient code instrumentation for transiently-powered embedded sensing. In *Information Processing in Sensor Networks (IPSN), 2017 16th ACM/IEEE International Conference on*. IEEE, 209–220.

[5] Alexei Colin, Graham Harvey, Brandon Lucia, and Alanson P Sample. 2016. An energy-interference-free hardware-software debugger for intermittent energy-harvesting systems. *ACM SIGPLAN Notices* 51, 4 (2016), 577–589.

[6] Alexei Colin and Brandon Lucia. 2016. Chain: tasks and channels for reliable intermittent programs. *ACM SIGPLAN Notices* 51, 10 (2016), 514–530.

[7] Alexei Colin, Emily Ruppel, and Brandon Lucia. 2018. A reconfigurable energy storage architecture for energy-harvesting devices. In *ACM SIGPLAN Notices*, Vol. 53. ACM, 767–781.

[8] Santosh K Gaikwad, Bharti W Gawali, and Pravin Yannawar. 2010. A review on speech recognition technique. *International Journal of Computer Applications* 10, 3 (2010), 16–24.

[9] Alessandro Giusti, Amy L Murphy, and Gian Pietro Picco. 2007. Decentralized scattering of wake-up times in wireless sensor networks. In *European Conference on Wireless Sensor Networks*. Springer, 245–260.

[10] Graham Gobieski, Brandon Lucia, and Nathan Beckmann. 2019. Intelligence Beyond the Edge: Inference on Intermittent Embedded Systems. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 199–213.

[11] Josiah Hester and Jacob Sorber. 2017. Flicker: Rapid prototyping for the batteryless internet-of-things. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. ACM, 19.

[12] Josiah Hester, Kevin Storer, and Jacob Sorber. 2017. Timely Execution on Intermittently Powered Batteryless Sensors. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. ACM, 17.

[13] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine* 29, 6 (2012), 82–97.

[14] Greg Hopper and Reza Adhami. 1992. An FFT-based speech recognition system. *Journal of the Franklin Institute* 329, 3 (1992), 555–562.

[15] Impinj Inc. 2018. Impinj Speedway R420 RFID Reader Product Information. https://www.impinj.com/platform/connectivity/speedway-r420/. Last accessed: Apr. 8, 2019.

[16] IXYS Corporation. 2018. IXOLAR™ High Efficiency SLMD121H04L Solar Module. http://ixapps.ixys.com/DataSheet/SLMD121H04L_Nov16.pdf

[17] JBL. 2019. JBL Go+ bluetooth speaker. https://www.jbl.com/

[18] Frederick Jelinek. 1997. *Statistical methods for speech recognition*. MIT press.

[19] Vincent Liu, Aaron Parks, Vamsi Talla, Shyamnath Gollakota, David Wetherall, and Joshua R Smith. 2013. Ambient backscatter: wireless communication out of thin air. In *ACM SIGCOMM Computer Communication Review*, Vol. 43. ACM, 39–50.

[20] Brandon Lucia, Vignesh Balaji, Alexei Colin, Kiwan Maeng, and Emily Ruppel. 2017. Intermittent Computing: Challenges and Opportunities. In *LIPIcs-Leibniz International Proceedings in Informatics*, Vol. 71. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[21] Brandon Lucia and Benjamin Ransford. 2015. A simpler, safer programming and execution model for intermittent systems. *ACM SIGPLAN Notices* 50, 6 (2015), 575–585.

[22] Amjad Yousef Majid, Delle Donne Carlo, Kiwan Maeng, Alexei Ytidtnm, Colin, Kasim Sinan, Przemystaw Pawetczak, and Brandon Lucia. 2019. Dynamic Task-Based Intermittent Execution for Energy-Harvesting Devices. *ACM Transitions on Sensor Network* (2019).

[23] Amjad Yousef Majid, Michel Jansen, Guillermo Ortas Delgado, Kasim Sinan Ytidtnm, and Przemystaw Pawetczak. 2019. Multi-hop Backscatter Tag-to-Tag Networks. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 721–729.

[24] Monsoon Solutions Inc. 2018. High Voltage Power Monitor. https://www.msoon.com/hvpm-product-documentation

[25] Saman Naderiparizi, Aaron N Parks, Zerina Kapetanovic, Benjamin Ransford, and Joshua R Smith. 2015. WISPCam: A battery-free RFID camera. In *2015 IEEE International Conference on RFID (RFID)*. IEEE, 166–173.

[26] pui audio, vesper. 2019. PMM-3738-VM1010-R: A ZeroPower Listening piezoelectric MEMS microphone. http://www.puiaudio.com/pdf/PMM-3738-VM1010-R.pdf

[27] Benjamin Ransford and Brandon Lucia. 2014. Nonvolatile memory is a broken time machine. In *Proc. MSPC*. ACM, Edinburgh, United Kingdom.

[28] Benjamin Ransford, Jacob Sorber, and Kevin Fu. 2011. Mementos: system support for long-running computation on RFID-scale devices. In *ACM SIGARCH Computer Architecture News*, Vol. 39. ACM, 159–170.

[29] VS Rao. 2017. Ambient-Energy Powered Multi-Hop Internet of Things. (2017).

[30] Saleae. 2017. Logic 16 Analyzer. http://www.saleae.com. Last accessed: Jul. 28, 2017.

[31] Joshua R Smith, Alanson P Sample, Pauline S Powledge, Sumit Roy, and Alexander Mamishev. 2006. A wirelessly-powered platform for sensing and computation. In *International Conference on Ubiquitous Computing*. Springer, 495–506.

[32] Vamsi Talla, Bryce Kellogg, Shyamnath Gollakota, and Joshua R Smith. 2017. Battery-free cellphone. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 2 (2017), 25.

[33] Texas Instruments. 2018. Ultra Low Power Management IC, Boost Charger Nanopowered Buck Converter Evaluation Module. http://www.ti.com/tool/BQ25570EVM-206

[34] Texas Instruments. 2019. MSP430FR5994 16 MHz Ultra-Low-Power Microcontroller Product Page. http://www.ti.com/product/MSP430FR5994

[35] Joel Van Der Woude and Matthew Hicks. 2016. Intermittent computation without hardware support or programmer intervention. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 17–32.

[36] Taras K Vintsyuk. 1968. Speech discrimination by dynamic programming. *Cybernetics and Systems Analysis* 4, 1 (1968), 52–57.

[37] Kasım Sinan Yıldırım, Amjad Yousef Majid, Dimitris Patoukas, Koen Schaper, Przemyslaw Pawelczak, and Josiah Hester. 2018. Ink: Reactive kernel for tiny batteryless sensors. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*. ACM, 41–53.

[38] Hong Zhang, Jeremy Gummeson, Benjamin Ransford, and Kevin Fu. 2011. Moo: A batteryless computational RFID and sensing platform. *University of Massachusetts Computer Science Technical Report UM-CS-2011-020* (2011).

[39] Yi Zhao, Joshua R Smith, and Alanson Sample. 2015. NFC-WISP: A sensing and computationally enhanced near-field RFID platform. In *RFID (RFID), 2015 IEEE International Conference on*. IEEE, 174–181.