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! The key idea being the use of multiple intermittent nodes to ensure that (at least) one is on at all times, providing a sense of continuous availability. To establish the required number of nodes, we modeled -and validated on real hardware- the specific (dis)charge behavior of individual nodes over time, and the emerging collective availability.

An important finding is that a CIS designed for certain (minimal) energy conditions requires no explicit spreading of awake times due to randomness in the power source and node hardware. 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 sleeping 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 90%.

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'19)*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

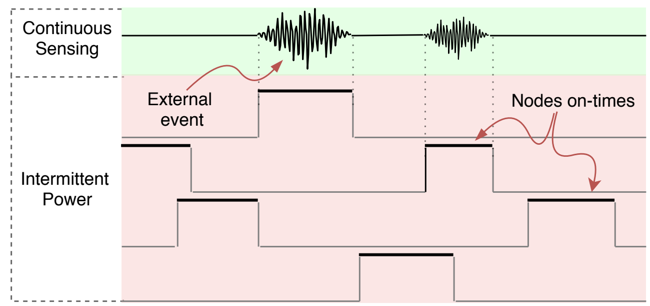


Figure 1: A Coalesced Intermittent Sensor (CIS) is a group of intermittently-powered nodes that sense continuously despite the intermittent power supply. CIS exploits the inherent randomization of energy harvesting systems, if available, and introduces artificial randomization, when needed, to preserve continuous sensing.

1 INTRODUCTION

The vision of smart cities, through the use of Internet of Things technologies, requires billions of sensors providing the necessary context to aid people in their daily lives. For example, cars will no longer need to wait in front of traffic lights for non-existing pedestrians to cross the road; doors, upon leaving, will provide people with the latest weather forecast; and jackets will adjust air circulation based on body temperature.

Unfortunately, batteries do not provide a viable solution to power all the needed sensors in smart cities. Batteries can be bulky, for example, 32% of a 1 g sensing "backpack" fixed on a cyborg insect is a battery [9]; hazardous; and expensive—they will deplete and require maintenance leading to service disruptions and heightened subscription fees. Moreover, the raw materials for making batteries are also limited. Therefore, future sensors must leave batteries behind and rely on green, perpetual energy sources.

Natural energy sources such as light, vibration, and heat can power tiny sensors directly [14–16, 30]. Tiny energy harvesters, however, can only scavenge a very limited power from such energy sources [25]. Therefore, an energy-harvesting sensor operates intermittently. An intermittent sensor starts by harvesting a certain amount of energy in its buffer (i.e., a super-capacitor). Subsequently, it triggers operation which depletes the buffered energy quickly, as the power consumption rate tends to be much higher than the power accumulation rate. Once the energy is below a certain level, the sensor experience a complete power-down, the cycle of charging and operating repeats [8].

Intermittent devices trade off a reliable energy source (the battery) for a sustainable energy source, i.e., ambient energy. This switch to energy harvesting generates many challenges [26]. For example, preserving computation progress under frequent power interrupts, enabling timely operations with indeterminate power-down duration, and the fact that nodes operate intermittently.

Researchers continue to investigate these challenges. For example, [2, 7, 26, 36, 37] studied the intermittent computation problem, which is concerned with the preservation of an application progress

Conference'19, July 2019, New York, NY, USA

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

and data consistency under frequent power failures; [18] investigated the timely operation challenge, which is concerned with data freshness after a power interrupt; [28] introduced a system design for peripherals states preservation for intermittently powered sensors; and [47] introduced event-driven execution for the intermittent domain, which deals with input and output operations under arbitrarily-timed power loss.

Despite significant progress achieved in the intermittent domain, *the system availability problem* has not been addressed. A monitoring sensor that has a very low probability to be available when an external event occurs is not worth deploying. A sensor that is capable of capturing only very short events has a limited number of potential applications. For example, a voice-controlled light-switch capable of only accepting short (single-word) commands has its limitations. Using "on" to turn on the lights might turn on other devices as well. Using "lights" does not allow the specification of "on" or "off". Consequently, intermittent sensors have not gained widespread adoption.

This paper tackles the paradox of continuous sensing on intermittent devices. It studies the randomized power cycle of energy-harvesting platforms and makes a key observation about the interrelationship between these platforms. Sensors driven by the same ambient energy source (e.g., light or RF) do *not* show correlated on/off (sense/charge) cycles. Building on top of this observation, the paper introduces a new type of sensor that we call *Coalesced Intermittent Sensor* (CIS). The CIS is defined as the abstraction of a group of energy-aware intermittent nodes providing the collective sense of being always on. Figure 1 illustrates the CIS concept for our prototype implementation of a command recognizer; a number of solar-powered nodes equipped with a microphone recording voice commands in a smart home setting. Recording and processing a word depletes the super capacitor powering a node, “silencing” it until the subsequent recharging completes. Multiple nodes with (partially overlapping) on/off cycles spread in time can provide continuous service despite the inherent intermittency.

In contrast to periodic (one-shot) sensing applications, event-based applications (like our command recognizer) may induce implicit synchronization (multiple nodes detecting the same command) that compromises the availability required by the application (all node recharge after the first word, missing any subsequent word). To guarantee continuous availability, a CIS may need to introduce artificial randomness. Doing so, however, is non-trivial as knowledge is needed about the number of (charged) nodes, which depends on a number of factors including environmental conditions regarding the power source (e.g., light intensity). This paper provides an estimator based on local measurements of a node's duty cycle that has been used effectively on our prototype command recognizer enabling it to detect commands of four words with above 90% detection accuracy. In summary this paper makes the following contributions:

- We introduce a new type of sensor that is an intermittently powered, yet senses continuously. The *Coalesced Intermittent Sensor* (CIS) is the abstraction of a group of intermittently-powered sensors exploiting (inherent) randomization to spread awake times uniformly.
- We model the (CIS) availability of energy-harvesting nodes and validate it against in-the-wild measured data and under controllable energy conditions.
- We introduce an algorithm for a node to determine its own duty cycle, which depends on the ambient power source. That duty cycle can effectively be used to estimate the number of active neighbors, which in turn decides if a node should back off to avoid duplicate event detection and availability interruptions (implicit synchronization in favourable harvesting conditions).
- We prototype, evaluate, and demonstrate the feasibility of the Coalesced Intermittent Sensor concept in the form of voice-control application recognizing individual words on solar-powered nodes equipped with a microphone.

2 RELATED WORK AND BACKGROUND

Recent advances in ultra-low-power microcontrollers along with the development of energy harvesters have enabled the creation of stand-alone battery-free sensors. Ambient energy that powers these sensors is volatile. Thus, these sensors operate intermittently.

2.1 Energy-harvesting systems

Ideally, energy harvesters 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 form a dedicated RF signal emitter, and solar power varies from tens of μW to tens of mW when it is harvested by a solar panels of a few cm^2 illumination surface [26, 38].

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 [40, 48, 50]; WISPcam, an RF powered camera [32] and, the battery-free cellphone [41]. Others, harvest from ambient sources such as the ambient backscatter tag [25], and the solar-powered tag [29]. Other platforms that facilitate the development of batteryless energy-harvesting systems have also been proposed. For instance, Fliker [17], a prototyping platform for batteryless devices; EDB [6] an energy-interference-free debugger for intermittent devices; and Capybara [8], 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 failure [4, 7, 13, 27, 44]. To this end, they decompose an application into several small pieces and save the state of the computation progress on the transitions between these code segments. Therefore, intermittent applications do not return to the same execution point (e.g., `main()`) after each power failure—in contrast to the applications that assume continuous power supply—instead they resume from the last successfully saved state of execution.

Intermittent systems are regarded as the successor of energy-aware systems. Dewdrop [5] is an energy-aware runtime for (Computational) RFIDs such as WISP. Before executing a task, it goes into low-power mode until sufficient energy is accumulated. QuarkOS [49] divides the given task (i.e., sending a message) into small segments and sleeps after finishing a segment for charging energy. However, these systems are not power disruption tolerant. In other words, if a system could not sustain the energy consumption of low-power-mode and powers down, then all the computation progress will be lost.

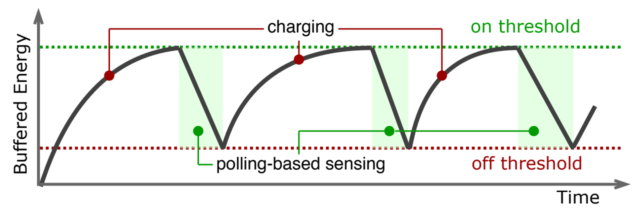
Mementos [37] proposed a volatile memory checkpointing-based approach to enable long-running applications on intermittently powered devices. DINO [36] enables safe non-volatile memory access despite power failures. Chain [7] minimizes the amount of data need to be protected by introducing the concepts of atomic tasks and data-channels. Hibernus [1, 2] measures the voltage level in the energy buffer to reduce the number of checkpoints. Ratchet [45] 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 [18] enables time-aware intermittent computing. InK [47] introduces event-driven intermittent execution. All the aforementioned studies focus their development on a single intermittent node. The paper is the first that considers the abstraction of a group of intermittent nodes and investigates the emerging collective duty cycle of the system.

2.3 Speech recognition

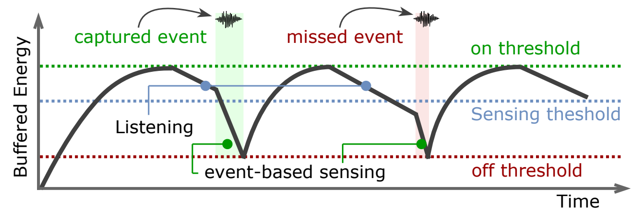
Speech recognition consists of several steps. The basic steps are: *Speech recording and signal digitization*—a microphone records the sound waves and an ADC converts the microphone signal into a digital signal. A sampling rate of about 8 kHz is required to capture the frequencies of a human voice (100-4000Hz [3]). *Framing*—after that the digitized signal is divided into blocks of usually 10-30 ms [10–12] called frames. *Features extraction*—for each frame a feature vector is extracted containing all the relevant acoustic information. *Feature matching*—finally the extracted features are matched against features known to the recognizer.

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) [46]. Approaches based on Hidden Markov Models showed much better performance than DTW-based ones [23]. Hence, they became the standard techniques for general-purpose speech recognition until artificial intelligent algorithms [19], however, outperform them. Furthermore, many specialized hardware architectures for speech recognition have been proposed to, for instance, reduce energy consumption [33, 34].

Speech recognition algorithms can be classified based on the type of speech that they can recognize into *spontaneous speech, continuous speech, connected word,* and *isolated word* [12]. Systems with *continuous* or *spontaneous speech* recognition are the closest to natural speech, but are the most difficult to create because they need special methods to detect words boundaries [12]. 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* type. It requires a period of silence on both sides of a spoken word and accepts only single words.



(a) When CIS does polling-based sensing, its energy consumption profile has, generally, two distinct rates: zero when it is charging, and a maximum when it is sensing.



(b) When CIS does event-based sensing, it stays in low-power mode waiting for an external event to wake up the node. Consequently, it has three distinct energy consumption rates: zero when it is charging, a maximum when it is sensing, and an in-between when it is sleeping. In favorable energy conditions, the sleeping mode may cause intermittent nodes to synchronize their power cycles on external events and miss the next ones.

Figure 2: Coalesced Intermittent Sensor (CIS) energy profile for different sensing strategies. Green bars highlight successful sensing operations while the red bar shows a failed sensing attempt due to insufficient buffered energy.

Voice is a natural way for the human to interact with small devices. However, implementing speech recognition on resources—memory, computation power, and energy—limited platforms is challenging, to say the least. Therefore, we attempt to recognize, with our command recognizer prototype, the simplest type of speech, isolated words.

3 COALESCED INTERMITTENT SENSOR

The Coalesced Intermittent Sensor (CIS) is the abstraction of a group of battery-less intermittent sensor nodes seeking to provide continuous availability to the user. The key to success is to exploit the properties (i.e. randomness) of the ambient energy source to arrive at a uniform spreading of the awake times of the individual senor nodes to achieve the maximum coalesced availability. To this end we will first analyze the energy-consumption life cycle of an intermittently-power node, as well as the joined availability of multiple nodes. Then we will point out a practical aspect that is often overlooked; when intermittent nodes operate in favourable ambient conditions, their uncorrelated behaviour changes due to

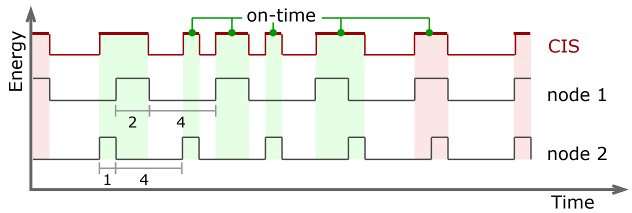


Figure 3: A Coalesced Intermittent Sensor's availability is the emerging collective on-time of its intermittent nodes' on-times. The difference between the power cycles leads to a constant relative shift between the nodes duty cycles. This, in turn, causes their on-times to be uniformly distributed on 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.

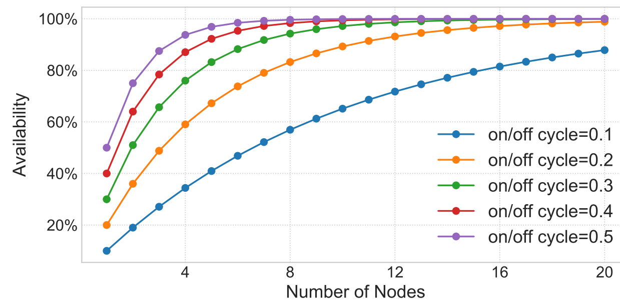


Figure 4: Coalesced Intermittent Sensor availability percentage for a different number of nodes and different duty cycles. The nodes are uniformly distributed and the CIS on-time evolves, when adding new nodes, according to the equation 1.

the extra energy leading to synchronized patterns that need to be scrambled by introducing artificial, controlled randomization.

3.1 Energy consumption

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 it activates its microcontroller for data acquisition and processing, see Figure 2a. (Note that we assume that the microcontroller is the dominant energy consumer module of a node.) 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. In case of the 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 hardware approach is most energy efficient, but can be mimicked in software by periodic polling (i.e. acquiring 1 sound sample and checking if it exceeds the acoustic noise floor). 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, see Figure 2b. Although in reality the sleep and active phases draw quite different levels of power (128 vs. 849 μW on our hardware, see Table 4) we will model the life cycle of a sensor node simply as an on-period followed by an off-period (in which the node recharges its energy buffer) as that suffices to analyze the collective availability of a CIS.

3.2 Coalesced availability

The CIS's on-time is the projection of its underlying intermittent nodes' on-times on the time axis. The CIS's on-time ranges from minimum (when all nodes on-times cluster together, see the red regions in Figure 3) to the maximum (when the overlapping between its nodes on-times is zero or when continuous availability is reached, highlighted with a green color in Figure 3). Two broad strategies for minimizing overlapping, hence, maximizing CIS availability, can be imagined:

i. *Explicit on-time division strategy*: Recent advancements in timing intermittent operations enable intermittent nodes to measure their on and off times with the help of an external ultra-low-power timer [18]. Similar breakthroughs in passive communication enable ultra-low-power message exchange between battery-less nodes [24]. Intermittent nodes can use these advancements to apply a time-division multiplexing strategy to explicitly avoid overlapping on-times. For example, a node calculates its average on-time $\overline{t_{on}}$ and off-time $\overline{t_{off}}$ for N power cycles. Then it measures the time difference between its power-up and the intended transmitting time Δt. Subsequently, it encodes the information $(\overline{t_{off}}, \overline{t_{on}}, \Delta t)$ in a message and broadcasts it. 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 either increasing (or decreasing) its power consumption to shorten (or lengthen) its on-time or shift its power cycle to a different time slot. This approach, obviously, assumes relatively stable nodes' power cycles. Assuming N nodes observe the same on -and off-periods, the coalesced availability (their collective on-time) would be N times the duty cycle of a node (or 100% if the total length of the on-times exceeds a single power cycle).

ii. *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 uniformly distribute the nodes' on-times is to ensure that their power cycles are different. This can be achieved by forcing intermittent nodes to go into low-power mode upon power-ups. The length of this mode is randomly chosen for each node. This will change the length of the nodes on-times and, consequently, alter their power cycles. Figure 3 shows the scenario of two intermittent nodes with different power cycles. Node 1 has a power cycle of 6 units of time and an on/off cycle of $\frac{1}{3}$. Node 2 has a power cycle of 5 units of time and an on/off cycle of $\frac{1}{5}$. Following the time axis from the left, we can see that the position of the on-time of Node 2 is shifted by 1 unit of time after each power cycle of Node 2. 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. If we extend the previous scenario to three or more nodes then the on-time of the resulting CIS can be described with the following formula,

$$t_{\text{on}}(N) = t_{\text{on}}(N-1) + \frac{t_{\text{off}}(N-1)}{t_{\text{off}}(N-1) + t_{\text{on}}(N-1)} \times t_{\text{on}}(1), \qquad (1)$$

where $N \in \mathbb{N}$ and $t_{\text{on}}(N)$ is the on-time of a CIS with N intermittent nodes. For the initial case where $N = 1$ we define $t_{\text{on}}(0) := 0$ and $t_{\text{off}}(0) := 1$.

In addition to characterizing the availability of a CIS, equation 1 also states that the benefit of adding a node to the CIS is proportional to the CIS's off-time. In Figure 4 CIS availability percentage for different duty cycles and different number of intermittent nodes are shown.

There is a clear trade-off between the aforementioned methods. While the explicit control method provides fine control over the system distribution and therefore requires less number of nodes than the implicit control method, the implicit control method does not depend on the ability to communicate between the nodes and therefore it is simpler and more energy efficient. Since inter-node communication is beyond the capabilities of most of today's intermittent nodes, we focus on the implicit approach.

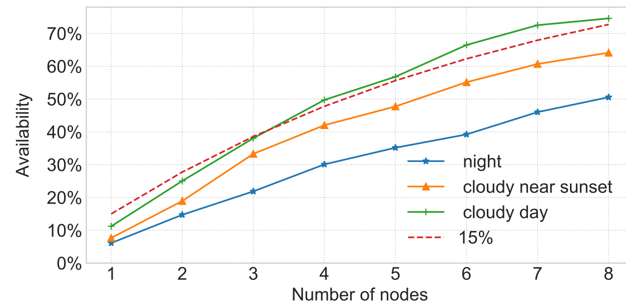
Implicit distribution of nodes' on-times. Figure 5 shows the availability of CISs when they are powered by different energy source and for a different number of intermittent nodes. We see that (i) the energy sources (sunlight, artificial light, and RF) power the CIS intermittently, (ii) the CIS availability increases with number of nodes, and (iii) this addition is proportional to the CIS off-time.

The dashed lines represent Equation 1 expectation about the CIS availability for certain power cycles (10% for the RF powered system, and 15% for the light powered one). By comparing these lines to the measured ones we can conclude that these energy sources provide sufficient randomness to cause each node to have a slightly different power cycle which, in turn, causes their uptimes to be uniformly distributed in time.

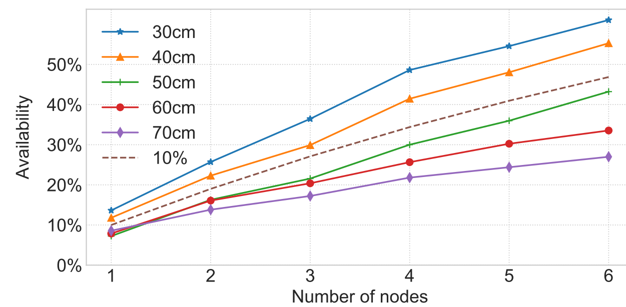
3.3 Power States

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 the 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 percentage (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.



(a) The system is powered by uncontrollable light sources—artificial light during the night, and sunlight during the day.



(b) The system is powered by radio frequency waves (RF). The RF source is located 30-70 cm aware from the RF tags (WISPs).

Figure 5: Measuring Coalesced Intermittent Sensor availability for a differed number of intermittent nodes. Generally, adding a node increases the system availability. This increment, however, is proportional to the CIS off-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 the darkness). In general, for under-targeted energy conditions, the CIS behavior can be considered as undefined.
- *Hibernating power state*—In event-based sensing scenarios, the intermittent nodes of a CIS sleep in low-power mode waiting for an external event to wake them up. If the energy conditions are relatively higher than the targeted conditions, the nodes may not die and sustain their sleeping power consumption. This will cause them to synchronize their wake-ups on the first incoming event and their power down as the event capturing process depletes their energy buffers quickly. Consequently, the CIS may miss the next incoming events (specially if the events happens to arrive in bursts) causing it to sense intermittently instead of continuously, see Figure 6.
- *Continuous power state*—Under direct mid-noon sun even a tiny solar panel can continuously power a sensor. In such conditions, the CIS will sense continuously without the need for randomization. Therefore, the job of a single node will be

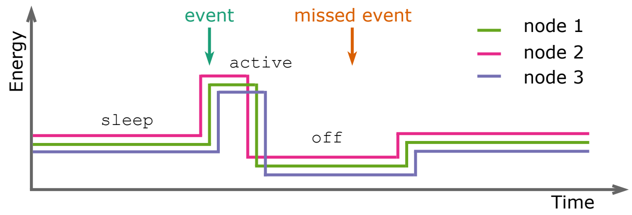


Figure 6: Coalesced Intermittent Sensor is in a hibernating power state when the energy harvesting rate approximates the energy consumption rate at the sleeping (or low-power) mode. In this state, the intermittent nodes lose the randomization in their power cycles. Thus, all the nodes capture the same event and power down shortly after missing the subsequent ones. Consequently, the CIS senses intermittently and does not take advantage of its redundant intermittent nodes to approach continuous sensing.

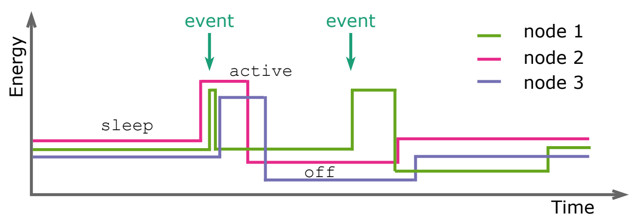


Figure 7: Randomized response helps in mitigating the hibernate-power-state problem. Also, it reduces the number of duplicated captured events when the CIS is overpowered. However, effective randomization strategy must be energy aware.

repeated N times, and instead of sending a single message to a battery-powered or tethered sink—to push the data to the internet—N identical messages will be sent which waste a lot of energy.

The inefficiencies highlighted in the hibernating and continuous power states can be mitigated by enforcing randomization on the response of intermittent nodes (Figure 7): 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. 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.

Choosing a fix response probability is an inefficient way of dealing with the over-powering problem as the number of active intermittent nodes at a given moment 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 at the moment of an external event arrival (which is discuss it next) and respond proportionally.

Algorithm 1 off-time estimation

1: $f_{\text{REBOOT}}(u) = u{+}{+}$ ▷ power reboot counter
2: $i \leftarrow f_{\text{REBOOT}}(i)$ ▷ i is a persistent variable
3: E_{buf} ▷ Size of energy buffer
4: t_a ▷ time of discharging E_{buf} at load a, no harvesting
5: $t_i \leftarrow x$ ▷ timing every x power cycles
6: **if** $(i \mod t_i) = 0$ **then**
7: $\quad i = 0$
8: $\quad f_{\text{LOAD}}(a)$ ▷ set node load to a
9: $\quad t_{\text{on}} \leftarrow t_{\text{PERS}}()$ ▷ persistent infinite loop
10: **end if**
11: **if** $i = 0$ **then**
12: $\quad \Delta t = t_{\text{on}} - t_a$ ▷ time difference due to charging
13: $\quad E_{\text{har}} \leftarrow (E_{\text{buf}} \div t_a) \times \Delta t$ ▷ harvested energy
14: $\quad P_{\text{in}} \leftarrow E_{\text{har}} \div t_{\text{on}}$ ▷ incoming power
15: $\quad t_{\text{off}} \leftarrow E_{\text{buf}} \div P_{\text{in}}$
16: **end if**

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

light intensity (lux)	mean	std
300	1.01	0.85
500	1.63	0.98
800	2.88	1.50
1200	5.05	1.08

3.4 Intermittent Timing

Timing is a key building block of sensing systems. However, it is missing on intermittent nodes unless an additional dedicated timer circuit is added to them [18]. Here we would like to propose an alternative way that does not require additional timer hardware. Obviously, the on-time of an intermittent node can be measured using the microcontroller's built-in timers. However, the difficulty is *how an intermittent node can time its own off-time?*. Actually, answering this question does not only enable timing on intermittent devices but also enables them to estimate the richness of the ambient energy.

Timing the death. Algorithm 1 shows how a node can estimate its off-time. The basic idea is that a node measures its on-time while harvesting (Line 9) and compares it to the time required to drain the super capacitor without charging. The additional on-time Δt is the result of the energy accumulated while executing. (Line 12). By assuming a relatively stable charging rate, a node can calculate how long it will be off charging (Line 13-15). Obviously, in order for the time estimation to be correct, the reference time and the on-time measurement must be done with same load (a).

3.5 Nodes overlapping

In order for a node to estimate the number of active nodes at a given moment, first, it has to know the total number of nodes (N) in its CIS, which we assume to be known to the nodes before deployment. Second, this analysis is built on the observation that a node's on-time is a good indicator about the on-times of other nodes in the CIS, see Figure 8. A node can measure its on-time t_{on} and off-time t_{off} using Algorithm 1 (or an external dedicated timer [18]). Then,

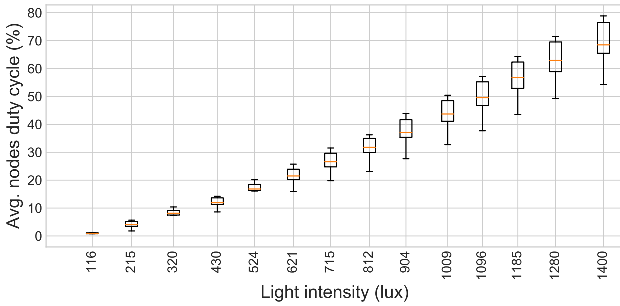


Figure 8: The average duty cycles of eight intermittently powered nodes for different light intensity. In general, an individual node's duty cycle is a good indicator of the average duty cycle of its CIS.

it 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

$$t_{\max} = N \times t_{\text{on}}. \tag{2}$$

Then, from Equation 1, the node calculates the CIS expected on-time ($t_{\text{on}}(N)$). As we argued in Section 3.2, nodes on-times are uniformly distributed over the CIS power cycle. Thus, the overlapping on-time is also uniformly distributed over the CIS on-time. Then, a node can calculate the average number of active intermittent nodes n_{active} using the following formula,

$$n_{\text{active}} = t_{\max} \div t_{\text{on}}(N). \tag{3}$$

and choose the proper randomization factor. If a second event, however, happens shortly after the first one, a node needs to update N as follows,

$$N = N - n_{\text{active}} - 1$$

the -1 is because the node itself decided not to react on the first event.

Table 1 shows the average number of overlaps of an 8-nodes CIS for different light intensities. These measurements validate that nodes overlapping time is uniformly distributed over the CIS on-time. For example, at 1200 lux an individual node of our CIS has a duty cycle of $\approx 62\%$. If we multiply it by the number of nodes (Equation 2) we get about 500%. Figure 4 indicates that a CIS with eight nodes of duty cycles above 50% has near 100% availability. From equation 3, we find that the expected number of clustered nodes is 5 which is what Table 1 also shows.

4 IMPLEMENTATION—COALESCED INTERMITTENT COMMAND RECOGNIZER

We have developed a prototype of a coalesced intermittent command recognizer (CICR): an instant of a Coalesced Intermittent Sensor. The CICR consists of eight batteryless intermittent nodes. Each node is capable of performing isolated words recognition.

The reason behind developing a CICR is threefold: (i) voice is a natural and convenient way for human to interact with miniaturized devices; (ii) demonstrating *the world's first* batteryless intermittently-powered command recognizer, which shades light on the potential of batteryless intermittent systems; and (iii) facilitating testing with different sensing strategies and different type of external events arrival (i.e., regular or burst).

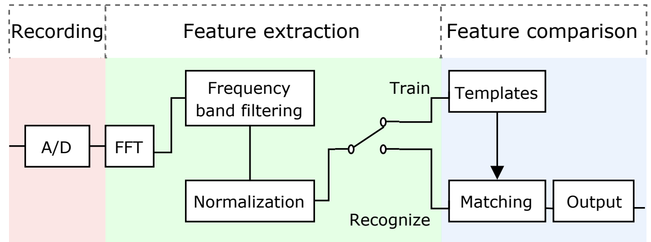


Figure 9: Coalesced Intermittent Command Recognizer: an instant of a Coalesced Intermittent Sensor. CICR features a power failure immune word recognition algorithm. First a word is recorded. Then, its spectral features are extracted. The resulting features vector is compared against previously-stored words' templates for recognition. The comparison is done using a liner distance matching algorithm

4.1 Hardware

A CICR node consists of thee main parts: a microphone, a microcontroller, and a harvester. MSP430RF5994 [43], 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 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 [35], 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 [42] connected to an IXYS SLMD121H04L solar cell [21] and a super-capacitor of 470 µF. For debugging we used the Saleae logic analyzer [39].

4.2 System Description

The CICR has a 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 9 and explained below:

4.2.1 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. To determine the end of the recording we relied on the characteristics of the targeted vocabulary. In particular, we identified experimentally the minimum effective recording length, which is 285 ms for the chosen set of words. By exploiting the Wake-on-Sound feature and using the minimum effective recording length, we eliminate the

Table 2: Profiling of features matching algorithms: Dynamic Time Warping (DTW) and Linear Distance Matching (LDM).

Section	LDM (ms)	DTW (ms)
Recording	285	285
Feature extraction	501	501
Feature matching	99	1251
Total	885	2037

need for an endpoint detection algorithm, greatly improving the processing time and system efficiency from the energy perspective.

4.2.2 Feature Extraction. Once a recording has finished, framing and data processing begin. CICR divides the digitized signal into non-overlapping frames of 256 samples (≈ 33 milliseconds). This size is beneficial for doing a Fast Fourier Transform and short enough for the voice-features to be considered constant inside a frame.

To extract the spectral features of a frame, CICR divides the frequency of interest into 12 bands (as in [20]). The first five bands have a bandwidth of 200 Hz. The next three have a bandwidth of 300 Hz which are followed by two bands of 500 Hz. Finally, the last two bands have a 600 Hz bandwidth. This division is motivated by how the energy is concentrated in human speech [20]. Then, CICR computes the 256-point Fast Fourier Transform for each frame. The resulting feature vector contains the amount of energy concentrated in each frequency band defined earlier. This feature vector forms the basis for the words identifying process once it is normalized.

We normalize the feature vectors to reduce detection errors that result from differences in the amplitude of the speech input. To normalize a feature vector, CICR computes the binary logarithm of each entry of that vector. Then it computes the mean of the resulting vector. Finally, it subtracts the computed mean from each entry of the resulting vector. This is summarized in the following equation:

$$f_i = \log(\hat{f}_i) - \frac{\sum_{i=1}^{S} \log(\hat{f}_i)}{S}, \tag{4}$$

where f_i is the normalized output for the i^{th} spectral band of a feature vector, $\hat{f}_i$ is the energy in the i^{th} spectral band of the frame, and S is the number of spectral bands (12 in our case).

4.2.3 Feature Matching. Feature matching is achieved by computing the distances between the normalized feature vectors of the recorded word and the feature vectors of the words stored during the training phase (templates). CICR computes the squared Euclidean distance between vectors as follows:

$$d_j = \sum_{i=1}^{S} (f_{s,i} - f_{r,i})^2, \tag{5}$$

where d_j is the distance between the j^{th} stored and recorded vectors. $f_{s,i}$ is the normalized output of the i^{th} spectral band of a stored vector, $f_{r,i}$ is the normalized output of the i^{th} spectral band of a recorded vector. The total distance between two words is calculated as follows:

$$D_k = \sum_{j=1}^{l} d(j) \tag{6}$$

where D_k is the distance between the k^{th} stored word and the recorded word, and l is the recording length measured in frames.

Once the recorded word has been compared to all CICR template words, the template with the smallest distance to the recorded word is considered the correct word. However, if the smallest distance is bigger the garbage threshold which we experimentally set, then the CICR will return "undefined word".

It should be emphasized that in the linear distance matching algorithm (LDM) the feature vectors of two words are compared successively, not accounting for differences in pronunciation speed. This is sufficient for our case as we are targeting isolated words and speaker dependent speech recognition type. We also implemented the Dynamic Time Warping algorithm which better handles the difference in the speed of speech. However, it is slower than the linear matching algorithm (Table 2) and the detection accuracy was comparable in our case. Therefore, we default our implementation to LDM.

4.2.4 Power Failure Protection. In order to preserve the progress state and to protect CICR data against randomly timed power failures, we manually split CICR program into 19 atomic regions. We ensured the each of these regions requires less energy than what the energy buffer can provide with a single charge. The program progress state is saved 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.

4.3 Code profiling

The entire command recognition software was written in the `C` programming language. The total program consists of 973 lines of code, excluding the FFT function from the Texas Instrument DSP library. See Table 3 for more information.

Table 3: Code statistics: lines of code

Language	Files	Blank	Comment	Code
C	7	264	173	736
C/C++ Header	8	62	40	237
Total	15	326	213	973

The memory footprint on the microcontroller is 20,064 bytes of FRAM and 1,134 bytes of SRAM. Execution times are shown in Table 2.

The power usage of a node differs according to it's activity. When a node is waiting for a voice event, it is in low-power mode. When data needs to be processed or recorded it is in active mode. When recording, the microphone and ADC consume additional power. The power consumption rates are determined by measuring the current with a Monsoon power monitor [31] and shown in Table 4.

Table 4: Power usage

Section	Current (μA)	Voltage (V)	Power (μW)
Sleeping	64 ±20	2.008	128 ±40
Recording	423 ±20	2.008	849 ±40
Processing	282 ±20	2.008	566 ±40

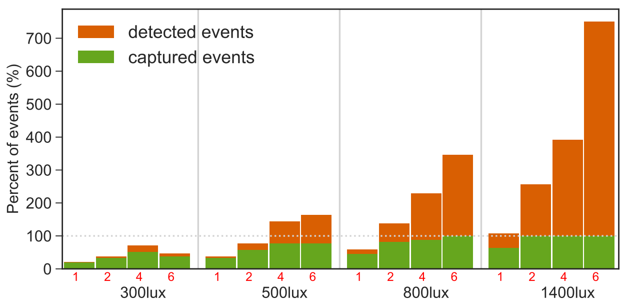


Figure 10: The number of detected and captured events by the coalesced intermittent command recognizer with eight intermittent nodes. The total number of external events is 240. In general, we see that when the light intensity increases, the number of detected and captured events rise too. Moreover, there is a positive correlation between the length of the inter-event arrival time and the detection and capture rates. Red numbers indicate events arrival interval.

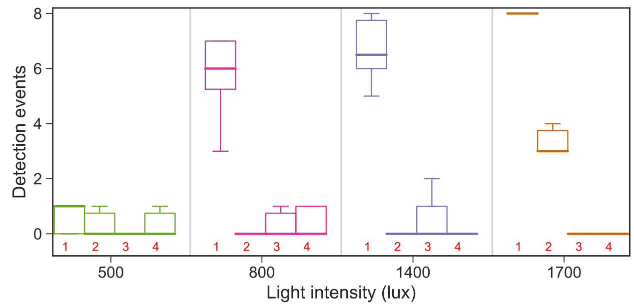


Figure 11: When capturing a burst of events without randomized response, the majority of the nodes react to the first event of a burst and power down, missing the rest of the burst. Red numbers indicate events index in a burst.

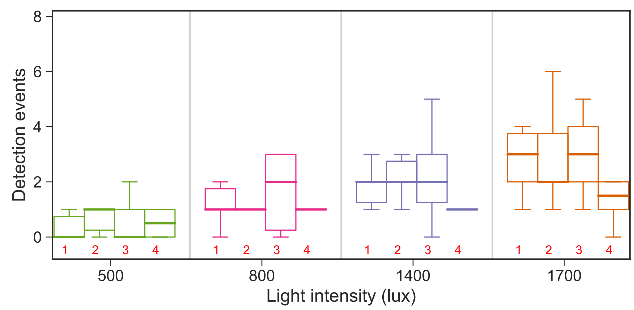


Figure 12: Response randomization enables a CIS to capture the entire burst of events with high capturing rates. It also reduces the number of duplicated events. Red numbers indicate events index in a burst.

5 EVALUATION

To evaluate the performance of the Coalesced Intermittent Sensor, we conducted several experiments in different energy conditions and with different types of events.

5.1 Experiment setup

After validating our observation on natural light and office artificial light, we designed a testbed with controllable light intensity for clarity and results reproducible. To this end, we blocked uncontrollable light sources with a box of 60×40 cm. On the ceiling of the box, we attached a light strip of 2.5 m with 150 LEDs that can produce 15 different light intensities. On the bottom of the box, we placed a coalesced intermittent command recognizer of 8 intermittent nodes (the hardware is described in Section 4.1).

The events in our experiments are spoken words (Table 6). We recorded different patterns of isolated words to emulate the arrival of bursts or individual events with varying inter-event and inter-bust timing. We used a Bluetooth speaker [22] to replay a certain record. The data were collected using logic anaylyzer [39] and processed on a laptop running Ubuntu 16.04 LTS.

5.2 Events detection rate

These experiments show the total number of detected events and the number of uniquely detected ones with and without randomization. Also, they were conducted for a different type of events.

Regular events. Figure 10 shows the percentage of the total captured events and the uniquely captured ones. In this experiment we vary the light intensity from 300 lux to 1400 lux and the inter-event time from 1 sec to 6 sec.

We clearly see a positive correlation between light intensity and the number of detected events. In particular, we see that the number of duplicated detected events rises dramatically when light intensity increases, demonstrating the overpowering problem. 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 sleep longer in low-power mode, and this reduces the inherent randomization of the intermittent nodes and leads them to the *hibernating power state* (Section 3.3).

Indirectly, these results show how a CIS can achieve a much higher duty cycle than its individual intermittent nodes—Figure 8 shows that with a light intensity of 800 lux an intermittent node is active with a duty cycle of 30% while Figure 10 shows that a CIS of 8 nodes captures 100% of the unique events when the time between them is 6 s.

Bursty events. Figure 11 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.

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

Table 5: Randomized response reduces the number of duplicated detected events, when the CIS is overpowered, by 50% while losing only 7% of the unique events. The results are presented in the following format *unique/total* detected events.

Each burst was repeated 10 times and for four different light intensities. The nodes sleep in low-power mode when they finish processing, waiting for the next event.

In general, we observe that the intermittent nodes react to the first event of a burst and power down shortly after missing other events in the burst. This results validate our theory about the side effect of the *hibernating power state* (Section 3.3). These results also demonstrate the hibernating power problem on a wide range of power intensities, showing the significance of this problem. Next, we will show how randomized response can mitigate these problems.

5.3 Events detection rate with randomization

Regular events. Table 5 shows the number of detected events for three different scenarios. We see that randomized response reduces duplicated events by an average of ≈50%, while only marginally lowers the number of the uniquely detected events. The intermittent nodes were responding to events with a probability of 65% for the scenario of 800 lux and 6 seconds arrival time and the scenario of 1400 lux and 4 seconds arrival time. However, for the highest energy level and the longest inter-event arrival time a responding probability of 30% was used.

Bursty events. Figure 12 shows that a CIS with randomized response spreads its resources–as compared to Figure 11–and captures the entire burst with a probability of above 90%. We also observe a positive impact of randomized response when the system is under-powered (500 lux).

To randomize during bursty events, a node reacts with a certain probability on a event. This probability is different for each event since the node become active after the last recharge. In order to spread the nodes over the events, the probabilities need to increase for subsequent events, since some nodes have reacted already on previous events, and therefore the number of nodes still available is smaller after each event. A node reacts with a probability of 40% on the first event, with 50% on the second event, 70% on the third event and 100% on the fourth event.

5.4 Coalesced intermittent command recognizer word detection accuracy

Table 6: Testing set

on	off	stop	clear	load
go	pause	resume	edit	cancel

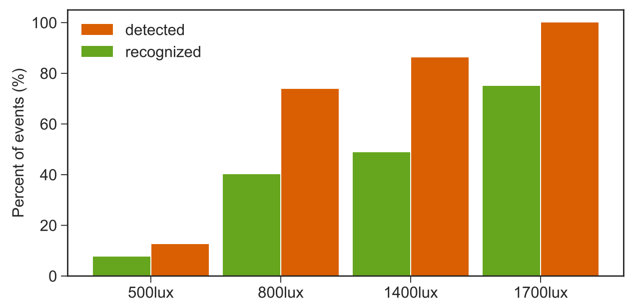


Figure 13: Average number of successfully recognized words per node and average number of detected words per node, as percentages of the total number of played words. Words are relatively long events and therefore some of their recordings do not complete due to insufficient harvested energy.

For evaluating the coalesced intermittent command recognizer accuracy, we used the word set in Table 6. Each word was pronounced by a single speaker 20 times and recorded on a PC. One of these recordings was stored as a template on the CICR, while the remaining 19 were played back through a Bluetooth speaker [35] for testing.

The ratio between detected events and successfully recognized events per node is shown in Figure 13 and it averages out at 76.7%. The difference between detection and capture is primarily caused by nodes that have insufficient buffered energy to finish recording.

6 DISCUSSION AND FUTURE WORK

Intermittent timing algorithm 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 external events in favorable energy conditions.

Coalesced Intermittent Sensor availability on a fine scale Because of the differences in intermittent nodes power cycles, their duty cycles are constantly shifting relative to each other (Figure 3). However, this study does zoom in on the CIS availability on short time scale and the effect of length of the differences between the nodes power cycles on the system short term availability.

Coalesced Intermittent Sensor Speech recognition on intermittent devices In this paper, we have shown the feasibility of speech recognition on intermittent power. We also demonstrated the possibility of recognizing burst of events (in our case four words). However, the type of speech we targeted is the simplest, isolated words. Next, we may attempt recognizing a more complicated type of speech and for a larger number of words than the number chosen for this study.

Additionally, the command recognition rate could further be improved by using an estimation of the energy left in the energy buffer, to start recharging early. This will prevent a detection when there is not enough harvested energy to record for a long enough time, letting a node recharge earlier and coming back with sufficient energy.

7 CONCLUSION

We presented the *Coalesced Intermittent Sensor* (CIS), an intermittently powered "sensor" that senses continuously! CIS is built around the observation that multiple intermittent nodes distribute themselves uniformly in time. This observation enables us to accurately model, and validate on real hardware, the CIS availability—the collective on-time of its intermittent nodes.

An important finding is that favorable energy conditions may cause sleeping intermittent nodes to synchronize their power cycles on the arrival of the first event. Consequently, they react to the same event, start recharging at the same time, and missing the next event. To counter this unwanted behavior we designed an algorithm to estimate the number of sleeping neighbors and respond proportionally to an event. We show that the Coalesced Intermittent Sensor is able to distribute bursts of events on its nodes "evenly" and capture the entire burst with above 90% detection accuracy.

Additionally, we prototype a battery-less coalesced intermittent command recognizer and show that it can successfully capture events of multiple words.

REFERENCES

[1] 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.

[2] Domenico Balsamo, Alex S. Weddell, Geoff V. Merrett, Bashir M. Al-Hashimi, Davide Brunelli, and Luca Benini. 2015. Hibernus: Sustaining Computation During Intermittent Supply for Energy-harvesting Systems. *IEEE Embedded Syst. Lett.* 7, 1 (March 2015), 15–18.

[3] C. Bernal-Ruiz, F.E. Garcia-Tapias, B. Martin-del Brio, A. Bono-Nuez, and N.J. Medrano-Marques. 2005. Microcontroller Implementation of a Voice Command Recognition System for Human-Machine Interface in Embedded Systems. *2005 IEEE Conference on Emerging Technologies and Factory Automation* 1 (2005), 587–591. https://doi.org/10.1109/ETFA.2005.1612576

[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] Michael Buettner, Ben Greenstein, and David Wetherall. 2011. Dewdrop: an Energy-aware Runtime for Computational RFID. In *Proc. NSDI.* USENIX, Boston, MA, USA, 197–210.

[6] 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.

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

[8] 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.

[9] Denis C Daly, Patrick P Mercier, Manish Bhardwaj, Alice L Stone, Zane N Aldworth, Thomas L Daniel, Joel Voldman, John G Hildebrand, and Anantha P Chandrakasan. 2010. A pulsed UWB receiver SoC for insect motion control. *IEEE Journal of solid-state circuits* 45, 1 (2010), 153–166.

[10] Brian Delaney, Nikil Jayant, Mat Hans, Tajana Simunic, and Andrea Acquaviva. 2002. A low-power, fixed-point, front-end feature extraction for a distributed speech recognition system. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, Vol. 1. IEEE, I—-793.

[11] Brian Delaney, N Jayant, and T Simunic. 2005. Energy-aware distributed speech recognition for wireless mobile devices. *IEEE design & test of computers* 22, 1 (2005), 39–49.

[12] 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.

[13] 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.

[14] Shyamnath Gollakota, Matthew S Reynolds, Joshua R Smith, and David J Wetherall. 2014. The emergence of RF-powered computing. *Computer* 47, 1 (2014), 32–39.

[15] Maria Gorlatova, Peter Kinget, Ioannis Kymissis, Dan Rubenstein, Xiaodong Wang, and Gil Zussman. 2010. Energy harvesting active networked tags (EnHANTs) for ubiquitous object networking. *IEEE Wireless Communications* 17, 6 (2010), 18–25.

[16] Maria Gorlatova, John Sarik, Guy Grebla, Mina Cong, Ioannis Kymissis, and Gil Zussman. 2014. Movers and shakers: Kinetic energy harvesting for the internet of things. In *ACM SIGMETRICS Performance Evaluation Review*, Vol. 42. ACM, 407–419.

[17] Josiah Hester and Jacob Sorber. 2017. Flicker: Rapid Prototyping for the Batteryless Internet-of-Things. In *Proc. SenSys.* ACM, Delft, The Netherlands.

[18] 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.

[19] 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.

[20] Greg Hopper and Reza Adhami. 1992. An FFT-based Speech Recognition System. *Journal of the Franklin Institute* 329, 3 (1992), 555–562.

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

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

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

[24] Jiangtao Li, Angli Liu, Guobin Shen, Liqun Li, Chao Sun, and Feng Zhao. 2015. Retro-VLC: enabling battery-free duplex visible light communication for mobile and IoT applications. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications.* ACM, 21–26.

[25] 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.

[26] 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.

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

[28] Kiwan Maeng and Brandon Lucia. 2019. Supporting Peripherals in Intermittent Systems with Just-In-Time Checkpoints. In *Proc. PLDI.* ACM, Phoenix, AZ, USA.

[29] Amjad Yousef Majid, Michel Jansen, Guillermo Ortas Delgado, Kasım Sinan Yıldırım, and Przemysław Pawełczak. 2019. Multi-hop Backscatter Tag-to-Tag Networks. *arXiv preprint arXiv:1901.10274* (2019).

[30] Robert Margolies, Guy Grebla, Tingjun Chen, Dan Rubenstein, and Gil Zussman. 2016. Panda: Neighbor discovery on a power harvesting budget. *IEEE Journal on Selected Areas in Communications* 34, 12 (2016), 3606–3619.

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

[32] Saman Naderiparizi, Aaron N. Parks, Zerina Kapetanovic, Benjamin Ransford, and Joshua R. Smith. 2015. WISPCam: A Battery-Free RFID Camera. In *Proc. RFID.* IEEE, San Diego, CA, USA, 166–173.

[33] Michael Price, James Glass, and Anantha P Chandrakasan. 2018. A Low-Power Speech Recognizer and Voice Activity Detector Using Deep Neural Networks. *IEEE Journal of Solid-State Circuits* 53, 1 (2018), 66–75.

[34] Michael Price, James R Glass, and Anantha P Chandrakasan. 2015. A 6 mW, 5, 000-Word Real-Time Speech Recognizer Using WFST Models. *J. Solid-State Circuits* 50, 1 (2015), 102–112.

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

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

[37] Benjamin Ransford, Jacob Sorber, and Kevin Fu. 2012. Mementos: System support for long-running computation on RFID-scale devices. In *Proc. ASPLOS.* ACM, Newport Beach, CA, USA, 159–170.

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

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

[40] Joshua R. Smith, Alanson P. Sample, Pauline S. Powledge, Sumit Roy, and Alexander Mamishev. 2006. A Wirelessly-Powered Platform for Sensing and Computation. In *Proc. UbiComp.* ACM, Orange County, CA, USA.

[41] 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.

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

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

[44] Joel Van Der Woude and Matthew Hicks. 2016. Intermittent Computation without Hardware Support or Programmer Intervention.. In *OSDI*. 17–32.

[45] Joel Van Der Woude and Matthew Hicks. 2016. Intermittent Computation Without Hardware Support or Programmer Intervention. In *Proc. OSDI*. ACM, 17–32.

[46] T. K. Vintsyuk. 1972. Speech discrimination by dynamic programming. *Cybernetics* 4, 1 (1972), 52–57.

[47] 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.

[48] Hong Zhang, Jeremy Gummeson, Benjamin Ransford, and Kevin Fu. 2011. *Moo: A Batteryless Computational RFID and Sensing Platform*. Technical Report UM-CS-2011-020. UMass Amherst.

[49] Pengyu Zhang, Deepak Ganesan, and Boyan Lu. 2013. Quarkos: Pushing the operating limits of micro-powered sensors. In *Proceedings of the 14th USENIX conference on Hot Topics in Operating Systems*. USENIX Association, 7–7.

[50] 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.