Continuous Sensing on Intermittent Power

Anonymous Author(s)

ABSTRACT

[to be rewritten]The vision of ubiquitous sensing runs into the reality of battery technology. Batteries are short-lived (about 14,000 tons of rechargeable batteries are thrown away in the United States alone); hazard; and costly—costs include manufacturing, replacement, and disposing. Batteryless sensors power themselves from ambient energy. Ambient energy is marginal and unpredictable. Consequently, tiny batteryless sensors operate intermittently, when sufficient energy, to do a simple task, is buffered, execution begins and terminates shortly after to recharge the depleted buffer. Sporadic sensing does not meet a wide range of real-world applications; therefore, intermittent sensors have not been widely adopted. We present a *Coalesced Intermittent Sensor* (CIS) an intermittent sensor that sense continuously! The CIS design takes advantage of the randomization embedded into the powering subsystem—an energy source and an energy harvester—to distribute its nodes uptime. Additionally, we realize the CIS in a prototype of a batteryless voice assistant agent. Our results show that x% of the commands are captured [uptime results].

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, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The Internet of Things is the engine that will drive smart cities. In these cities, cars will not need to wait in front of traffic lights for non-existing pedestrians to cross the road; doors, upon leaving, will provide people with weather forecast; and jackets will adjust air circulation based on body temperature. Smart cities will gain their awareness through billions of sensors.

Battery-powered sensors do not provide a viable solution to power all the sensors in smart cities. Batteries require (i) regular maintenance, even rechargeable ones wear out in a few years [**?**]; and (ii) hazard waste management. Moreover, the raw materials for making batteries are also limited. Therefore, future sensors must leave batteries behind and rely on perpetual energy sources.

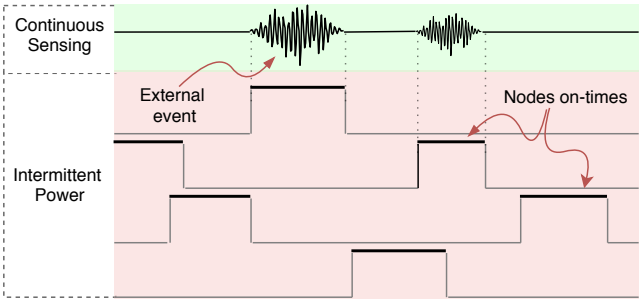


Figure 1: Coalesced Intermittent Sensor (CIS) is a group of intermittently powered sensory nodes that sense continuously despite the intermittent power supply. CIS exploits the inherent randomization of energy harvesting systems or enforce artificial randomization to preserve an efficient and continuous sensing functionality.

Natural energy sources such as light, vibration, and heat can power tiny sensors directly. Tiny energy harvesters, however, can only scavenge a very limited power from such energy sources. 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). Then, 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 continues indefinitely.

Intermittent devices trade-off a reliable energy source (the battery) for sustainable—when a large number of sensors are considered—energy source (ambient energy). This trade-off generates many challenges. For example, preserving computation progress under frequent power interrupts, enabling timely operations with indeterminate power-down duration, and the fact that nodes are not always available.

Many of these challenges have been tackled. For example, [4, 20, 21] studied the intermittent computation problem, which is concerned with the preservation of an application progress and memory integrity under frequent power failures; [8] investigated the timely operation challenge, which is concerned with data freshness after a power interrupt; and [**?**] introduced the event-driven execution for the intermittent domain, which is concerned with input and output operations under arbitrarily-timed power loss.

Despite the significant progress that has been 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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

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

© 2019 Association for Computing Machinery.

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

limited number of potential applications (imagine that you want to control room lights with a batteryless microphone. The microphone is capable of processing a single word. If you say "on" lights turned on but other systems might start to operate also—a specification problem. If you say "light" to eliminate other systems you lose the ability to control the light—a functionality problem). Consequently, intermittent sensors have not gained widespread adaptation.

This paper tackles the paradox of continuous sensing on intermittent devices by introducing a new type of sensors that we call *Coalesced Intermittent Sensor* (CIS). The CIS is defined as a group of intermittent nodes with randomized on/off cycles. CIS distinguishes between different energy conditions and adapts its response accordingly to efficiently distribute its resources and to approach continuous sensing. We put our observations and theory into test by realizing a CIS instant in the form of a distributed intermittent voice assistant agent. We tested the voice assistant in different energy conditions and the results validate our assumptions and observations.

Highlights of the paper contributions:

- [We introduce a new type of sensor that is intermittently powered, yet continuously senses.]
- [We modeled the CIS availability and validated it against on-a-real-hardware measured data.]
- [We introduced *intermittent timer* an algorithm that enables intermittent nodes to self-time their power cycles, without the need for extra hardware.]
- [We studied nodes overlapping when the power cycles of the intermittent nodes change.]
- [[maybe we skip this one]We proposed and algorithm for CIS event-based sensing.]
- [We prototype Coalesced Intermittent Sensor in a form of coalesced intermittent command recognizer.]
- Introducing a new sensor type: *Coalesced Intermittent Sensor* (CIS) is a group of intermittently powered nodes that take advantage of the randomized nature of the powering subsystem (energy source and energy harvester) to approach 100% system availability. CIS adapts its behavior based on the harvested energy conditions to preserve efficient and continuous operations.
- Characterizing the behavior of Coalesced Intermittent Sensor under different power arrival rates. We define four CIS operational states and investigate the associated operational challenges of these states.
- Characterizing the behavior of Coalesced Intermittent Sensor under different event occurrence frequencies and patterns.
- Implement a Coalesced Intermittent Sensor in the form of a distributed intermittent voice assistant agent. Voice control is a convenient interface for a human to interact with miniaturized devices. Moreover, it enables us to easily study the behavior of the CIS under a different type of external events (i.e. bursty arrival of events).

[Potential applications, relocate to the appreciated location] It increases the temporal and spatial availability of an intermittent system and enables resource distribution such as a large number of words templates for spoken words recognition systems.

Controlling the on/off cycle of intermittent devices enables adapting them to many real-world applications. For example, once a certain on/off cycle is preserved, an intermittent wake-up receiver can be implemented; an intermittent acoustic monitoring system for monitoring engines modules—the sound produced by a deformed gear tooth—can be made. Moreover, with the advances in passive communication (such as passive light [], and backscatter tag-to-tag [] communication) battery-free miniaturized sensors can form self-powered wireless sensor network to, for instance, create smart wallpaper and revolutionize smart buildings.

2 RELATED WORK AND BACKGROUND

2.1 Energy-harvesting

Ambient energy is volatile, and scarce. For example, radio waves harvestable power varies from nW-scale when harvesting ambient RF energy to µW-scale when harvesting a dedicated RF signal; and solar power ranges from tens of µW to tens of mW when it is harvested by a solar panels of a few cm^2 illumination surface [13, 22].

A tiny energy-harvesting device slowly charges its energy buffer (e.g. capacitor) while the device is off. Once the buffer is full, the device begins operating and depleting its energy reservoir—since power consumption is much higher than harvested power—until it powers down. This charging-discharging cycle repeats indefinitely.

Many batteryless energy-harvesting platforms have been proposed, for example Wireless Identification and Sensing Platform (WISP) [24] and its variants such as NFC-WISP [34], WISPcam [16], and NeuralWISP [30], batteryless phone [25], ambient backscatter tag [12] and Moo [32]. However, there is not 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 experience.

2.2 Intermittent execution

Intermittent execution models enable applications to progress despite frequent power failure [2, 4, 14, 28]. They decompose an application into several small code pieces and save the progress state of the application 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, as the applications that assume continuous power supply, instead they resume from the last saved progress state of execution. Intermittent systems are regarded as the successor of energy-aware systems. Dewdrop [3] is an energy-aware runtime for (Computational) RFIDs such as WISP. Dewdrop goes into low-power mode until sufficient energy for a given task is accumulated. QuarkOS [33] 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.

Mementos [21] proposed a volatile memory checkpointing-based approach to enable long-running applications on intermittently powered devices. DINO [20] enables safe non-volatile memory access despite power failures. Chain [4] minimizes the amount of data need to be protected by introducing the concepts of atomic tasks and data-channels. Hibernus [1] measures the voltage level in the energy buffer to reduce the number of checkpoints. Ratchet [29]

uses compiler analysis to eliminate the need of programmer intervention or hardware support. HarvOS [2] uses both compiler and hardware support to optimize checkpoint placement and energy consumption. Mayfly [**?**] enables time-aware intermittent computing. InK [31] introduces event-driven intermittent execution. All the studies in the intermittent domain have accepted the intermittent behavior of energy harvesting nodes as a fact and have not attempt controlling it. Our study is the first one that targets the intermittent systems availability problem and attempts to provide a continuous sensing despite the intermittent power supply.

2.3 Speech recognition

The speech recognition problem has been tackled from many angles and has experienced many great breakthroughs. For example, Dynamic time warping (DTW) algorithm enables matching voice signals with different speed (or time) []. Approaches based on Hidden Markov Models showed much better performance than DTW-based ones [11]. Hence, they became the standard techniques for general purpose speech recognition until artificial intelligent algorithms [9], however, outperform them.

Many specialized hardware architectures for speech recognition have been proposed to, for instance, reduce energy consumption [17, 18].

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* [7].

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 [7]. 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. This paper introduces the world first isolated words recognizer on energy harvested intermittent devices.

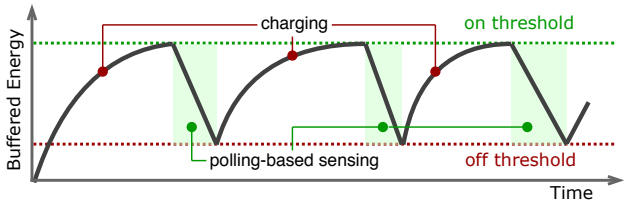
[relocate the following paragraph] Speech recognition consists of several steps. The basic steps are mentioned briefly here: *Speech recording and signal digitization*—a microphone records the sound waves and an ADC converts the microphone signal into a digital signal. A sampling rate of about 8 kHz is required to capture the frequencies of a human voice (100-4000Hz [**?**]). *Framing*—after that the digitized signal is divided into blocks of usually 10-30 ms [5–7] 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.

3 COALESCED INTERMITTENT SENSOR

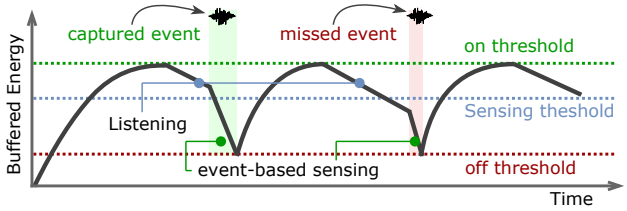
Coalesced Intermittent Sensor (CIS) is the abstraction of a group of batteryless intermittent sensors. CIS seeks to offer continuous sensing despite relying on ambient energy: an unpredictable and marginal power source.

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 a maximum when it activates its microcontroller for data acquisition and processing, (see Figure 2a)—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 listen) for an external event to wake up the microcontroller. This is important to minimize the energy wasted on the listening and to maintain the required energy budget for sensing for the longest possible time (Figure 2b). The idle listening mode, which is important for successful sensing, complicates the design of the CIS.



(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—staying in low power mode listening for an external event to happen—, its energy consumption profile has three distinct rates: zero when it is charging, a maximum when it is sensing, and an in-between energy consumption rate when it is listening. The third rate requires special attention when designing a CIS.

Figure 2: Coalesced Intermittent Sensor 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.

3.2 Availability

The CIS's on-time is the projection of its underlying intermittent nodes' uptimes 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 uptimes is zero or when the continuous time is reached). Two broad controlling strategies for minimizing nodes on-times overlapping and maximizing CIS's time span 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 [**?**]. Similar breakthroughs in

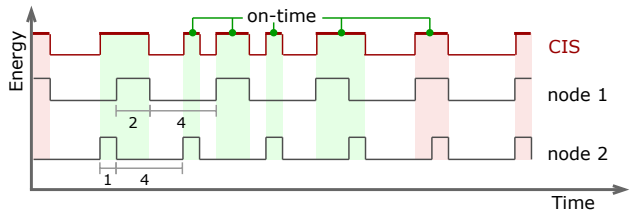


Figure 3: Coalesced Intermittent Sensor's on-time is the projection of its intermittent nodes uptimes on the time axis. When the power cycles of the intermittent nodes are different their uptimes distribution approaches uniform distribution. The red bars indicate a minimum CICR time span—CICR's nodes are overlapping—whereas the green bars show the maximum time span of the CICR.

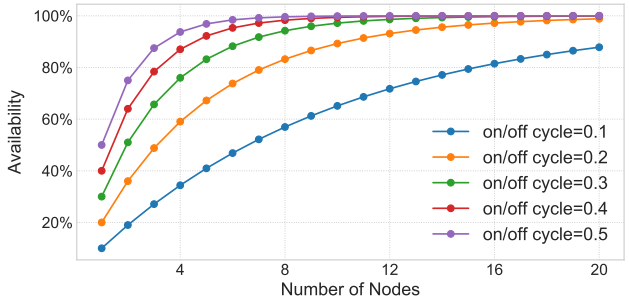


Figure 4: Coalesced Intermittent Sensor availability percentage for a different number of nodes and different duty cycles. The nodes are uniformly distributed and the CIS on-time evolves according to the equation 1 when new nodes are added.

passive communication enable ultra-low-power message exchange between batteryless nodes [**?**]. Intermittent nodes can use these recent advancements to apply time division multiplexing strategy to explicitly avoid nodes on-times clustering. 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. Then it encodes these information $(\overline{t_{off}}, \overline{t_{on}}, \Delta t)$ in a message and broadcasts it. If a node receives this message it will have full information about the transmitting node 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 and shift its power cycle to a different time slot. This approach, obviously, assumes relatively stable nodes power cycles.

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,

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

where $t_{on}(N)$ is the on-time of a CIS with N intermittent nodes. Figure 4 shows the CIS availability percentage for different duty cycles and different number of intermittent nodes.

There is a clear trade-off between the aforementioned methods. While the explicit control method provides fine control over the system distribution, the implicit method does not suffer from control messages exchange overhead. Although the implicit method is relatively simple to implement and explore, the explicit control method is not a far fetched idea considering the recent advancements in passive communication and intermittent timing. However, we opt to explore the implicit distribution control method as the hardware used to demonstrate the feasibility of passive light communication and ambient RF backscattering are not open source and re-making it is beyond the scope of this study.

3.3 Intermittent Timer

Timing is a key building block for sensing activities. However, it is missing on intermittent nodes, unless additional dedicated timer circuit is added to the intermittent nodes [**?**]. Here we would like to propose an alternative way the does not require additional timer hardware. Obviously, the on-time of a node can be measured using the build-in timers in the microcontroller. 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 it can also be used as a metric to estimate the environment energy richness. [it is also can be used to estimate how much energy is left the buffer at a certain moment while a node is on and has a stable power consumption]

Timing the death. Algorithm 1 shows how a node can estimate its off-time. The high level idea is that a node measures its on-time (Line 8) and compares it a reference time. The additional Δt is caused due to harvested energy while executing (Line 10). By assuming a relatively stable charging rate, a node can calculate how long it will be off charging (Line 11-13). Obviously, in order for the

Algorithm 1 off-time estimation

```
1:  f_REBOOT(u) = u++                          ▷ power reboot counter
2:  i ← f_REBOOT(i)                            ▷ i is 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 ← x                                    ▷ timing every x power cycles
6:  if (i mod t_i) = 0 then
7:      f_LOAD(a)                              ▷ set node load to a
8:      t_on ← t_PERS()                        ▷ blocking persistent timer
9:  end if
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
```

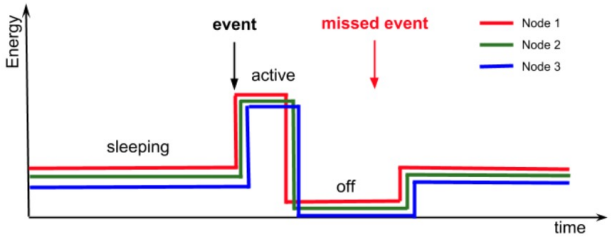


Figure 5: [Placeholder] No randomization

time estimation to be correct, the reference time and the on-time measurement must be done with same load.

3.4 Power States

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

(1) *Targeted energy conditions*—These are the energy conditions that the CIS designed for. In these energy conditions, the CIS should work intermittently and have sufficient power cycles randomness to uniformly distribute its intermittent nodes on-times and meet the desired availability percentage 4. In general, the targeted energy conditions should be near worst energy harvesting conditions to ensure that the system is properly functioning for the majority of the time.

(2) *Under-targeted energy conditions*—Ultimately, the ambient energy is an uncontrollable power source, and it is not hard to imagine scenarios where a CIS will be underpowered 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.

(3) *Hibernating energy conditions*—when CIS is tasked with event-based sensing, and the energy conditions are relatively higher than the targeted conditions, then the harvested energy may compensate for the energy consumption of the low-power mode. Thus, the system loses its randomness feature, and all the nodes will react to the first external event and power down shortly after as the energy consumption of sensing and processing is much higher than the energy consumption of the low-power-mode idle listening. If the external events happened to be sporadic, then this is not a big issue. However, for a bursty type of events (i.e. a command of two or three words), this will prevent the CIS from meeting its goal: continuous sensing (Figure 5).

(4) *Continuously powered*—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 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.

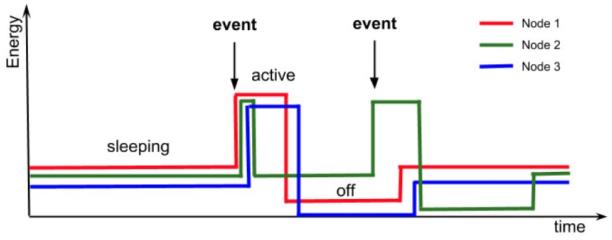


Figure 6: [Placeholder] Randomization

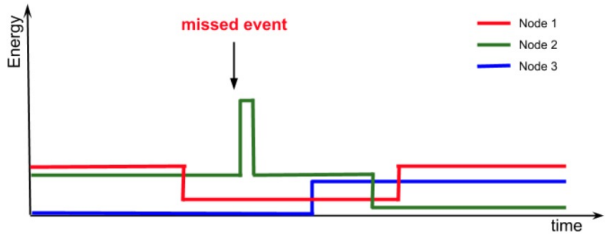


Figure 7: [Placeholder] Randomization side effect

The problems mentioned in (3) and (4) can be mitigated by enforcing randomization on the response of the intermittent nodes (Figure 6): when a node is wakened 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 (Figure 7). Therefore, the CIS has to distinguish between the targeted and above-targeted energy conditions and apply response randomization only during the above-targeted energy conditions. To do this, an intermittent node can measure the energy intensity using a dedicated ultra-low-power sensor or using the on-time duration as a metric to estimate the ambient energy intensity. A node energy buffer holds a certain maximum amount of energy, know during manufacturing. This energy powers the node for a certain amount of time t_{nopwr} for a fixed load l_{fix}. t_{nopwr} can

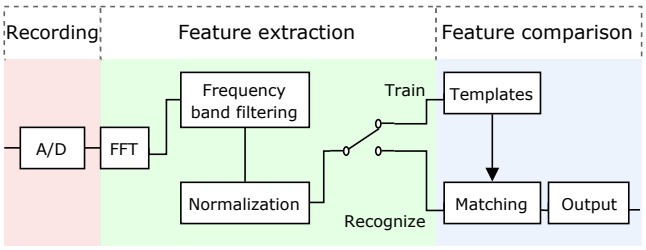


Figure 8: Coalesced Intermittent Command Recognizer: an instant of a Coalesced Intermittent Sensor. CICR features a power failure immune word recognizer. Once a word is recorded, the word's spectral features extraction begins. The resulting features vector is compared against previously-stored words' templates for recognition. The comparison using a liner distance matching algorithm

be embedded in the intermittent node memory before deploying. When the node wants to measure the incoming power, it set its load to l_{fix} and measure its on-time t_{pwr}. The time difference between t_{pwr} and t_{nopwr} can be used as a metric to estimate the incoming power and choose the response probability accordingly.

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).

Moreover, we believe that a CICR can facilitate direct human-to-human or human-to-objects communication. Imagine that a CICR based system is deployed in a play ground, embedded in ground and other objects. You want to call your child, and a CICR based system is embedded in your shirt and his shirt. You say his name and the CICR picks up the word and scatter it over light—[**?**] demonstrates the feasibility of scattering sunlight to communicate between two nodes that are up to 60 m apart. The scattered signal will be relied on other CICR nodes until the receiving node on his shirt receives the message and notify him, daddy is calling.

4.1 Hardware

A CICR node consists of thee main parts: a microphone, a microcontroller, and a harvester. MSP430RF5994 [27] 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 [19], 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 [26] connected to an IXYS SLMD121H04L solar cell [10] and a super-capacitor of 470 µF. For debugging we used the Saleae logic analyzer [23].

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 8 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 7,812 Hz. 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 happened be 259 ms for the chosen set of words. By exploiting the Wake-on-Sound feature and the minimum effective recording length we eliminate the 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 [**?**]). The first five bands has a bandwidth of 200 Hz. The next three has a bandwidth of 300 Hz which is followed by two bands of 500 Hzs. Finally, the last two bands has a 600 Hz bandwidth. This division is motivated by how the energy is concentrated in human speech [**?**]. 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.

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

$$f_i = \log(\hat{f}_i) - \frac{\sum_{i=1}^{S} \log(\hat{f}_i)}{S}, \quad (2)$$

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 done by computing the distances between the normalized feature vectors of the recorded word and the vectors of the words stored during the training phase. CICR computes the squared Euclidean distance between vectors as follows,

$$d_j = \sum_{i=1}^{S} (f_{s,i} - f_{r,i})^2, \tag{3}$$

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:

$$D_k = \sum_{j=1}^{l} d(j) \tag{4}$$

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 dictionary words, the word with the smallest distance to the recorded word is considered the correct word. However, if the smallest distance is bigger the garbage threshold [Partick uses confusion matrix for setting this threshold. Maybe I need to consider adding it] which we experimentally set, then the CICR will return "undefined word".

It should be emphasized that in Linear distance matching (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[1].

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 splitted CICR program into 19 atomic regions. We ensured the each of these regions requires less energy the 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 (`mian()`) after each power failure. Data in the non-volatile memory with Write-After-Read dependency are double buffered to ensure the 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 Texas Instrument DSP library, from which the FFT function was used. See 1 for more information.

The memory footprint on the microcontroller was 20,064 bytes of FRAM and 1,134 bytes of SRAM. Execution times are shown in 4.

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 measured with a Monsoon power monitor [15] and shown in 2.

[1]We have also implemented Dynamic Time Warping algorithm which better handles the difference in the speed of speech. However, it is slower than the linear matching algorithm and the detection accuracy were comparable in our case

Table 1: 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

Table 2: 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

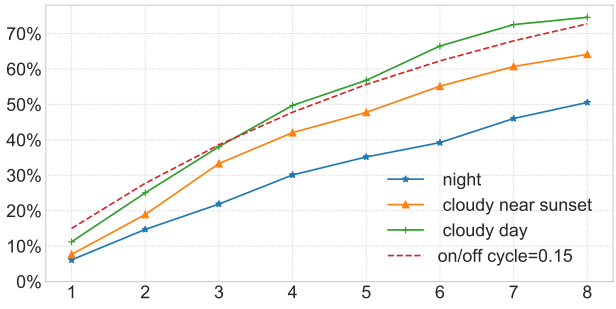


Figure 9: Implicit Coalesced Intermittent Sensor' nodes distribution using solar power harvesting randomization

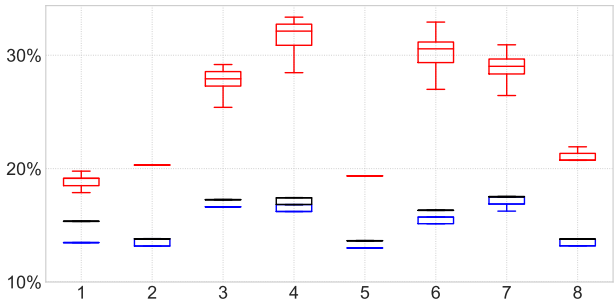


Figure 10: [Placeholder] Nodes duty cycles

5 RESULTS

Light-based Implicit Nodes' Uptimes Control. As Figure 9 shows light powers tiny sensory nodes intermittently. It also indicates that CIS availability increases as more intermittent nodes added. The dashed line is the simulated time coverage of the system as the intermittent nodes power-ups are uniformly distributed. By comparing the measured CIS's availability line graphs to the simulated one we can conclude that harvesting solar power of artificial light provides sufficient randomness to distribute the coalesced nodes on-times uniformly. Figure 10 shades more light on this observation. It show that each node has it own unique duty cycle. The little differences between the duty cycles causes them to have a "near" constant shifts relative to each other which in turn causes the system to nodes' on-times to have near uniform distribution.

Table 3: Testing set

on	off	stop	clear	load
go	pause	resume	edit	quit

5.1 Effective recording length

Since our target hardware has extremely limited resources, the first experiment targets the minimum effective recoding length without significant accuracy loss.

For this experiment a single microcontroller running on continuous power is used. Each word from Table 3 was recorded on a PC

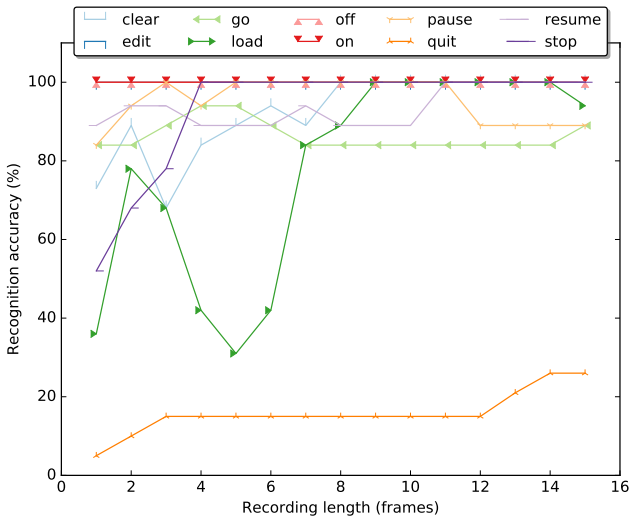


Figure 11: Recognition accuracy versus the recording length (in frames), while using multiple recordings per word for testing. Used feature matching method: linear.

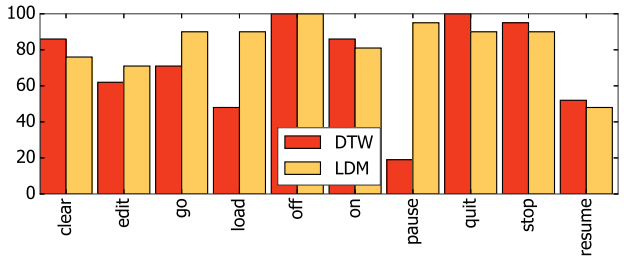


Figure 12: Recognition accuracy for linear matching and DTW when using 9 frames as the recording length.

20 times. The features—normalized FFT-based values—of the first recording are saved in the microcontroller's persistent memory as a signature to perform the feature matching, during the testing. the rest of the recording are used for conducting the experiment.

Figure 11 shows words recognition accuracy when the 19 recordings of each word are played back from the PC speaker. We can concluded that recording beyond *nine frames* do not increase the recognition accuracy; therefore, nine frames recording length is chosen for the rest of the experiments.

5.2 Comparison of feature matching methods

We implemented two algorithms for voice features matching, Dynamic Time Warping (DTW) and Linear Distance Matching (LDM). Due to the microphone wake-on-sound feature and the fixed recording length, DTW did not outperform LDM as Figure 12 shows. Moreover, DTW takes more time to process that data as Table 4 shows. Therefore, LDM algorithm is used in future experiments.

Table 4: Profiling of features matching algorithms: Dynamic Time Warping (DTW) and Linear Distance Matching (LDM)

Section	Linear (ms)	DTW (ms)
Recording	285.9	285.9
Feature extraction	501.9	501.9
Feature matching	99.4	1251
Total	887.2	

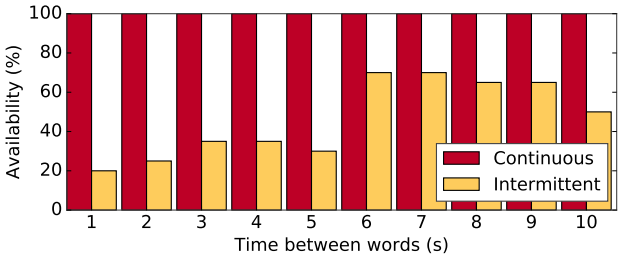


Figure 13: The effect of the time between consecutive words on the availability: the percentage of words that is processed by the command recognizer. [Add correct recognition as stacked bar?]

Table 5: The mean and standard deviation of the parameters that were used to simulate intermittent execution. Here t_{on} is the time the device is on while recording or processing, t_{off} is the time the device is charging and t_{sleep} is the time the device is sleeping while waiting on sound input.

Parameter	μ (ms)	σ (ms)
t_{on}	590	17.7
t_{off}	5310	154
t_{sleep}	22420	652

5.3 Intermittent microphone availability

This experiment shows the effect of intermittent power supply on a microphone availability [and recognition accuracy] for different command (or word) repetition speed. Each word (from Table 3) was played back 20 times with intra-words-playing time ranges from 1 to 10 seconds.

For the experiment we chose to simulate the intermittent power supply, based on a real measurement, to ensure environment consistency across different words. In particular, we measured the t_{on}; used an on/off ratio of 10% to simulate harsh harvesting conditions [21**?**]; and calculated the t_{sleep} based on the micro controller specification [**?**] (Table 5).

The obvious observation from Figure 13 is that intermittency has a great impact on the command recognizer availability. However, a less obvious, but important, observation is that the correlation between the on/off cycle and the command repetition speed has also great impact as the bars labeled with 5 and 6 from Figure 13 indicate.

6 DISCUSSION AND FUTURE WORK

7 CONCLUSION

REFERENCES

[1] 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.

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

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

[12] 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.

[13] 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.

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

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

[16] 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.

[17] 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.

[18] 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.

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

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

[21] 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.

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

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

[24] 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.

[25] 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.

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

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

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

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

[30] Daniel J Yeager, Jeremy Holleman, Richa Prasad, Joshua R Smith, and Brian P Otis. 2009. Neuralwisp: A wirelessly powered neural interface with 1-m range. *IEEE Transactions on Biomedical Circuits and Systems* 3, 6 (2009), 379–387.

[31] 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.

[32] 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.

[33] 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.

[34] 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.