Experimental Study of Lifecycle Management Protocols for Batteryless Intermittent Communication

Vishal Deep*, Mathew L. Wymore*, Alexis A. Aurandt, Vishak Narayanan, Shen Fu, Henry Duwe, Daji Qiao
Electrical and Computer Engineering, Iowa State University, Ames, IA, USA
{vdeep, mlwymore, aurandt, vishakn, shenfu, duwe, daji}@iastate.edu

***Abstract*—Batteryless energy-harvesting sensor nodes can operate indefinitely, but if the harvesting rate is too low, they must operate *intermittently*. Intermittent operation imposes various challenges upon the system. One of the least-studied is communication—if nodes are unpowered for long, unpredictable periods of time, how can they reliably communicate with each other? In prior work, we proposed the concept of lifecycle management protocols (LMPs) to mitigate this issue and enable wireless communication directly between intermittent sensor nodes using active radios. In this paper, we propose a design framework for a class of LMPs. We then provide analytical models for the delay and throughput of two-node communication using this framework. Finally, we implement this framework on hardware and validate our models in an experimental setting. To the best of our knowledge, this is the first design framework for, and implementation of, protocols for enabling and improving general-purpose communication between intermittent sensor nodes using active radios.**

***Index Terms*—Lifecycle Management Protocols (LMPs), Batteryless, Energy Harvesting, Intermittent Communication**

I. Introduction

Ubiquitous sensing could revolutionize a broad array of applications, from infrastructure management to public health to disaster mitigation. However, traditional deployments of sensor networks suffer from the battery problem: battery-powered devices have a finite lifetime, and when the battery dies, replacing it is costly and in some environments, dangerous or prohibitive. Sensor nodes can use a variety of energy harvesting techniques to replenish their energy supply over time, but matching the energy needs of a node with harvesting typically requires significant physical space, as well as cost. Additionally, traditional high-energy harvesting techniques such as solar are severely compromised or unavailable in some environments (indoors, in the dark, embedded in concrete, etc.). Finally, batteries themselves have longevity and robustness concerns [1], as well as environmental concerns regarding manufacturing and end-of-life processes [2].

Recent research has thus considered replacing the battery of a sensor node with a small energy buffer, such as a small capacitor (e.g. 50 mF or less). In conjunction, the energy requirements for harvesting in these batteryless systems are relaxed—instead of harvesting with the goal of maintaining an energy balance and remaining constantly on, the node harvests energy with the goal of charging its capacitor so that it can *eventually* turn on. Fig. 1 illustrates this *intermittent* operation. When the voltage on the capacitor reaches the *on threshold*, the node turns on and operates for a limited time (T_{ON}), until the energy store can no longer support operation. Then the node will *die*, with the CPU becoming completely unpowered, and it will remain off until the energy harvesting is able to recharge the capacitor to the on threshold again (T_{OFF} in the figure). We refer to this repeated on-time/off-time pattern as *lifecycling*.

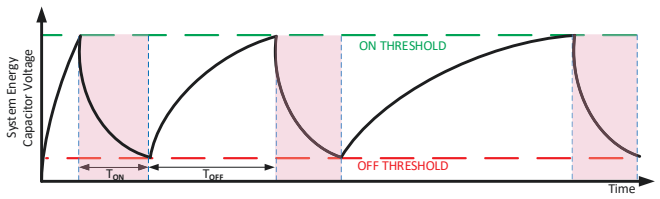


Fig. 1: Illustration of lifecycling behavior of an intermittent sensor node. As shown here, the available harvesting power can change over time, resulting in varying off-times.

Batteryless intermittent nodes could be small, inexpensive, robust, and effectively zero-maintenance, and therefore could be deployed at scale even in harsh and inaccessible environments. However, the challenges faced by intermittent systems are clear. Intermittent nodes lose all volatile state when they die. They have trouble keeping track of time across off-times. And, of particular interest in this work, direct communication between intermittent nodes is especially challenging. Not only do both nodes' wireless radios need to be on (as in a traditional duty-cycled wireless communication problem), but *the nodes themselves* need to be on at the same time in order to achieve communication. We refer to such a time as *overlap*. For intermittent nodes, "natural" overlap may be very rare—for example, a node's on-time may be hundreds of milliseconds at most, while its off-time may easily be tens of seconds, or minutes or hours in a very low-energy environment.

In prior work [3], we have proposed the concept of *lifecycle management protocols* (LMPs) for communication directly between batteryless intermittent nodes using active radios. These protocols can influence the lifecycling of a node to improve overlap and present more communication opportunities. In this paper, we explore this idea in more depth by proposing and implementing a general framework for the design of *stateless* LMPs. By stateless, we mean these LMPs do not retain information across off-times, making them simpler than stateful LMPs and a good starting point for implementation. Specifically, our contributions are:

- We propose the first framework to reason about and evaluate the LMP design space, focusing on stateless

*The first and second authors contributed equally to the paper.

LMPs.

- We propose formal metrics to quantify the constraints intermittency imposes on communication delay and throughput.
- We present analytical models to quickly estimate these metrics for communications between two intermittent nodes using our LMP framework.
- We present the first physical implementation of (stateless) LMPs used for active communication between two nodes powered solely by RF harvesting. We use this implementation to experimentally validate our analytical models.

II. Background & Related Work

Intermittent systems are an emerging research area, and to this point, there is little work on active communication directly between intermittent nodes. Our related work discussion first focuses on approaches to low-energy communication at the physical and MAC layers that may be relevant to intermittent systems. We then present background information on LMPs.

A. Physical Layer Approaches

Significant work in wireless sensor networks (WSNs) and the Internet of Things focuses on how to improve the energy efficiency of communication at the physical layer. One popular approach is backscatter modulation with reflective radios, including conventional RFID (reader-to-tag) backscatter communication [4], [5] and ambient backscatter communication [6], [7]. Instead of generating its own RF carriers as in traditional modulation, a backscatter transmitter modulates and reflects received RF signals, saving energy. Backscatter is traditionally used for a single hop between a "tag" and an always-on sink. "Tag-to-tag" backscatter communication also has been studied [8], [9], but only with nodes powered by constant sources or larger-scale energy harvesting.

In general, this type of physical layer technology is orthogonal to the intermittent node communication problem we study in this paper (i.e., it could be used in conjunction with an LMP to improve energy efficiency). That said, we design our LMPs with traditional active radio communication in mind, with the intent of enabling general-purpose wireless communication between intermittent nodes, even when powered solely by small-scale energy harvesting such as RF.

B. MAC Layer Approaches

Wireless sensor networks have a rich history of research into MAC layer approaches for reducing energy consumption in communication, most notably through *duty-cycled* MAC protocols (e.g. [10]–[12]). In these protocols, nodes turn their radios off when not communicating in order to conserve energy. At first glance, this appears conceptually similar to the lifecycling we study in intermittent nodes. However, in duty-cycling, only the radio is cycled, while the rest of the system (including the CPU, timers, etc.) remains powered. This means the node can duty-cycle at will, whereas an intermittent node in off-time is entirely subject to the unpredictable and uncontrollable charging rate. We therefore emphasize that lifecycling is a new and distinct problem for wireless communication.

Another approach that has been studied for WSNs is wakeup radios (WuR) [13], [14], where nodes operate a secondary, very-low-power radio that listens for a wakeup signal from a transmitter. When a wakeup signal is received, the node turns on its main radio for communication of data. Like duty-cycled MACs, WuR schemes assume some portion of the system is powered, so lifecycling is a separate problem. Still, as with the physical layer approaches, both WuRs and duty-cycled MACs could be used in conjunction with an LMP to further improve communications in intermittent nodes.

C. Lifecycle Management Protocols

LMPs manipulate the lifecycling of an intermittent node in order to maximize the utility of the limited energy available to the node. The concept of LMPs was introduced in [3]. [15] explores a similar concept of aligning overlap in the context of a neighbor discovery protocol, whereas we focus on general-purpose communication.

We define the *lifecycle ratio* of an intermittent node as:

$$L = \frac{T_{\text{ON}}}{T_{\text{ON}} + T_{\text{OFF}}} \tag{1}$$

This describes the balance between the node's power consumption and its ability to harvest energy from its environment. For a given system, a larger lifecycle ratio means a more energy-rich environment.

For our baseline system, described in more detail in Section V, the on and off thresholds from Fig. 1 are fixed by a voltage supervisor. The size of the system energy store (assumed here to be a capacitor) is also fixed. When the on threshold is reached, the capacitor has a fixed amount of energy stored. Therefore, the (maximum) on-time T_{ON} of the system is determined primarily by the size of the capacitor and the power consumption of the system in operation. The charging rate also affects the maximum on-time, but in our target scenario, the charging power is dominated by the on-time power consumption of the system. On the other hand, the off-time T_{OFF} of the system is determined by the charging power, which is often unpredictable and uncontrollable. Eq. (1) is thus a dynamic quantity.

In order to manage lifecycling, LMPs need a control mechanism. In our work, we use a mechanism we call *early-die*: during on-time, a node can turn itself off *before* voltage reaches the off threshold. This shortens the on-time and saves energy, effectively shortening the following off-time (fluctuations in charging rate aside), because not as much charging is required to reach the on threshold again. This lifecycling control mechanism can be found in off-the-shelf hardware and is available on our implementation platform. Other potential control mechanisms include a dynamic on threshold [16], [17] or a dynamically-sized energy store [18], [19].

The LMP should be part of a comprehensive energy management solution for the entire system (communication, sensing, and computation). In this work we focus on communication, where the LMP can be conceptualized as an additional layer between the network and data link layers in

the network stack. Using whatever mechanism is available, the LMP controls the lifecycle as best as possible to maximize communication opportunities. Due primarily to fluctuations in charging rate, this control is fuzzy at best. However, simulation results have shown that even a simple LMP could greatly improve communication performance [3]. In this paper, we verify these results both analytically and on hardware.

In general, LMPs can be classified as either *stateless* or *stateful*. Stateful LMPs keep track of protocol information, such as timing information, across off-times. However, measuring off-times requires a specialized timekeeping mechanism known as a persistent clock [20]–[22]. In contrast, stateless LMPs do not track information across off-times, and are generally simpler to design and implement. A stateful LMP could potentially outperform a stateless LMP [3], but we concentrate on stateless LMPs in this work as our first step in formalized design and implementation on devices.

III. Stateless LMP Framework

In this section, we first present our framework for stateless LMP design, and then propose formal metrics for the evaluation of this framework and LMPs in general.

A. Framework

We propose a framework that defines a general format and set of behaviors for stateless LMPs. To the best of our knowledge, this is the first general framework to aid the design and analysis of protocols that manage lifecycling of intermittent nodes for direct node-to-node communication.

Our framework assumes the availability of the early-die control mechanism discussed in Section II-C. With this mechanism, the decision of the LMP is *when* to early-die. This results in an on-time for the node somewhere between a $\tau_{\min}$, determined by the boot time of the node and the minimum time required to attempt communication, and $\tau_{\max}$, determined by the on-time power consumption P_{ON} of the node, the size of the energy store, and the on threshold.

As shown in Fig. 2, in our stateless LMP framework, each node chooses an on-time $\tau \in [\tau_{\min}, \tau_{\max}]$. For each on-time beginning at $t = 0$, the LMP schedules an early-die operation at $t = \tau$ and then allows the MAC to attempt communication. The details of communication are left to the MAC layer, preserving the network stack model. If the node is not communicating at $t = \tau$, the early-die operation is executed and the node turns itself off. But if the node is communicating at $t = \tau$, the early-die operation is cancelled and the node remains on to maximize the communication opportunity; it early-dies when the communication is finished (either because all packets for this neighbor have been sent/received, or because a node is running out of energy). This scheme prioritizes successful communication over LMP operations—after all, the purpose of this framework is to facilitate communication.

Nodes using our framework need to select a τ. A key finding from [3] is that, ignoring boot overhead, L from Eq. (1) is constant across all choices for τ. Put another way, ignoring boot overhead, *early-dying does not waste energy*. We explore the design space for τ in detail, including the effect of boot overhead, in Section IV.

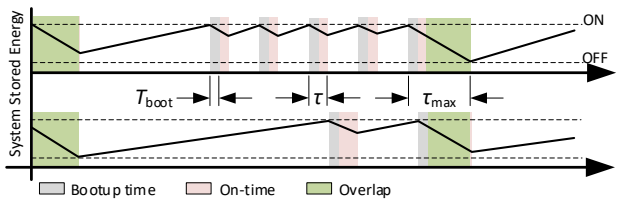


Fig. 2: Illustration of the proposed stateless LMP framework. Each node selects an on-time of length $\tau \in [\tau_{\min}, \tau_{\max}]$. This choice affects the node's frequency of on-times. When overlap (i.e. successful communication) occurs, the nodes extend their on-times to maximize the communication opportunity.

B. Metrics

As with any protocol, an LMP should be designed with a goal. However, we must first define metrics for measuring communication performance of an LMP. From the communication perspective, intermittency *constrains* performance—within a period of overlap, intermittency has no effect on communication performance. The purpose of the LMP is then to relax the constraints of intermittency as much as possible. We thus define two metrics:

1) **TTO:** time-to-overlap, the time between successive instances of overlap. This metric measures the communication delay introduced by the intermittency of the devices. This measures only time between overlap, the domain of the LMP. The MAC or physical layers may introduce additional delay in communication.
2) **OPS:** amount of overlap per second. This metric measures intermittency's constraint on the throughput of the communication channel. OPS measures overlap at the LMP level. Given this much overlap, throughput is further limited by the MAC, the physical layer, etc.

These metrics are related, but they are not equivalent. A short TTO may be useful in achieving a high OPS, but it is also possible for nodes to reach a high OPS by overlapping less frequently but for longer periods of time.

IV. Analytical Models

LMPs within our proposed framework are parameterized simply by the selected on-time τ. In this section, we analyze how the performance (TTO and OPS) of LMPs in our stateless framework is affected by the selection of τ. We focus our analysis on a two-node case. The intuitive tradeoff is that longer on-times provide a higher probability of overlap for any given on-time, while shorter on-times can happen much more frequently. Compounding the decision, τ could be the same for both nodes (*symmetric* on-times), or it could be longer on one node and shorter on the other (*asymmetric* on-times). Finally, the best value for τ may change depending on the application's needs. Therefore, we present analytical models that explore the design space of our framework (selection of τ) in terms of expected TTO and expected OPS.

A. Assumptions and Simplifications

To streamline the models, we assume that the on-time power consumption P_{ON} for a node is constant, regardless of whether the node is booting or running the radio, and that P_{ON} is the same for all nodes in the network. Similarly, we assume τ_{max} is the same for all nodes in the network. We also assume boot time T_{boot} is constant and the same for all nodes in the network, and that communication cannot occur until the boot process (both hardware and software) has finished. Since our target metrics measure only overlap, we do not model communication issues at the MAC or physical layers.

Our models consider a particular on-time of a node A. At a high level, we assume p, the probability that overlap occurs during a given on-time of node A, is independent of all other on-times. More specifically, as shown in Fig. 3, we consider some interval of length T_B (the average lifecycle period of a node B) that contains this on-time of A. We discretize this interval into slots of fixed length s. Node A's on-time begins at some slot (i.e., the *rebirth slot*) within this interval. For our models, we assume that node A's rebirth slot is randomly and uniformly distributed within the interval T_B.

In theory, the probabilities of overlap with a node B during on-times for node A are not independent. As an extreme example, if the charging rates of nodes A and B are constant, then the time between overlaps is predetermined and the probability of overlap at an on-time depends on the time since the last overlap. However, in our target scenario, the charging rates can fluctuate unpredictably, and these fluctuations can affect off-times significantly (with respect to the length of an on-time). This makes the assumption of independence reasonable in practice, a claim supported by the experimental results in Section VI.

B. TTO Model

In this section, we model the expected TTO of the design space of our stateless LMP framework in a two-node case. Our approach is to first model p, the probability of overlap for any given on-time of one of the nodes. Using p, we then model the expected number of on-times before overlap, which we directly translate to the expected TTO.

1) Probability of Overlap: We first discretize the time domain into slots of fixed length s. For logical convenience, we make s the duration of a single communication attempt, e.g. the time required to transmit a packet and receive an acknowledgement (or time out). We model an on-time of length τ as a sequence of $\frac{\tau}{s}$ slots, with the first b of those slots spent booting the node. These b slots are thus not available for communication and cannot count as overlap.

Consider any given on-time for a node A, and an interval of length T_B encompassing that on-time, as shown in Fig. 3. As T_B is the average length of B's lifecycle period, we assume an on-time for node B falls somewhere within this interval. As discussed in the assumptions, we assume the probability of B's rebirth slot being a particular slot is uniformly distributed within this interval. We can then model the probability of overlap with node B in any given on-time for a node A by counting the number of B's rebirth slots for which A and B have overlap, and dividing this quantity by the number of slots in T_B. As illustrated in Fig. 3, the sum of this count is $n+m-2b-1$, where m is the number of slots in B's on-time and n is the number of slots in A's on-time.

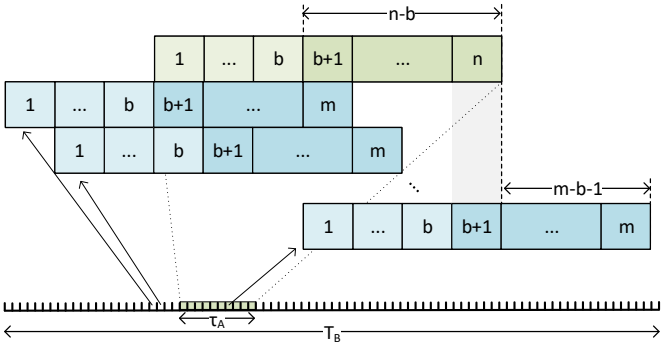


Fig. 3: Consider any interval T_B, divided into slots, surrounding an on-time for node A (green, shown at the top). With our model's assumptions, somewhere within this interval, node B (blue) has an on-time, with equal probability that it begins in any given slot. This figure enumerates the rebirth slots for B that produce overlap with node A. In the leftmost rebirth slot for node B that produces overlap, the final slot of B's on-time (slot m) overlaps with the first non-boot slot of node A (slot $b+1$). In the rightmost rebirth slot that produces overlap, the first non-boot slot of node B (slot $b+1$) aligns with the final on-time slot of node A (slot n). All rebirth slots between these two slots produce overlap, for a total of $n+m-2b-1$ rebirth slots with overlap.

The total number of slots in the interval is $\frac{T_B}{s}$. Therefore, the probability of overlap for any given on-time of node A is:

$$p = \frac{n+m-2b-1}{\frac{T_B}{s}}. \tag{2}$$

We can express T_i, the average lifecycle period for a node i, as a function of P_{ON}, τ_i, and the node's average charging power P_i, as follows:

$$T_i = T_{\text{ON}} + T_{\text{OFF}} = \tau_i + \frac{P_{\text{ON}}\tau_i - P_i\tau_i}{P_i} = \frac{P_{\text{ON}}\tau_i}{P_i}, \tag{3}$$

where the off-time is expressed as the energy consumed during on-time (minus the energy harvested during on-time) divided by the charging rate. Substituting Eq. (3) into Eq. (2) and translating number of slots to continuous time, we obtain:

$$p = \frac{P_B(\tau_A + \tau_B - 2T_{\text{boot}} - s)}{P_{\text{ON}}\tau_B}. \tag{4}$$

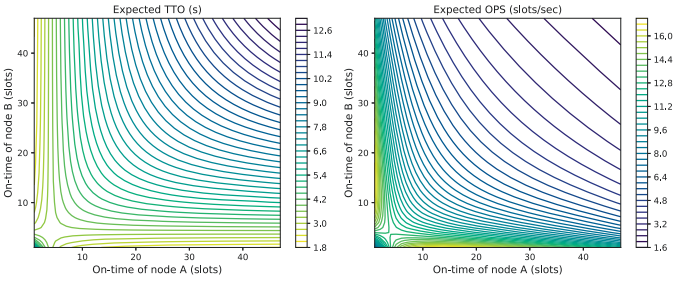
2) Expected Number of On-times until Overlap: Under our assumptions, we can model the on-times of node A as a series of independent Bernoulli trials with probability of success p. If we define X as the number of on-times until the next overlap, then X is a geometrically-distributed random variable with parameter p. Thus, from the geometric distribution, the expected number of on-times until overlap is:

$$E[X] = \frac{1}{p} = \frac{P_{\text{ON}}\tau_B}{P_B(\tau_A + \tau_B - 2T_{\text{boot}} - s)}. \tag{5}$$

3) Expected TTO: We model the expected time to overlap as the expected number of on-times of node A until overlap, multiplied by node A's lifecycle period T_A from Eq. (3):

$$E[\text{TTO}] = E[X] \cdot T_A = \frac{{P_{\text{ON}}}^2 \tau_A \tau_B}{P_A P_B (\tau_A + \tau_B - 2T_{\text{boot}} - s)}. \quad (6)$$

We plot the expected TTO versus values of τ_A and τ_B as a contour map in Fig. 4a. Our first observation is that the contour map is symmetric across the positive diagonal. This means that A and B can be swapped in the model, with the same results—even if the nodes have different lifecycles. In fact, examining Eq. (6), we observe that the charging rates of the two nodes (P_A and P_B) are simply scaling factors for the expected TTO. This is a striking finding—it implies that values for τ_A and τ_B that minimize expected TTO are the same across all charging rates for the two nodes. This suggests that *an LMP using our framework does not need to adapt to the lifecycle ratios of the two nodes* in order to maintain expected optimal performance.



(a) Expected TTO. (b) Expected OPS.

Fig. 4: Contour plots of our modeled metrics vs. τ for nodes A and B, with lifecycle ratios of 13.3% and 7.6% respectively (to match Section VI). On-time shown on the axes is in number of 5 ms slots and does not include boot time. Other parameters are set to match the experimentally-measured values described in Section V.

Returning to Fig. 4a, the best performance (smallest TTO) is in the upper left and lower right corners, where one node has an on-time of τ_{min} and the other has an on-time of τ_{max}. For symmetric on-times, around 5 slots (25 ms) provides the best performance. Intuition might suggest that τ_{min} would be the optimal symmetric on-time, but it is not, due to the T_{boot} term. Boot time is an overhead—it is on-time where energy is being consumed but overlap cannot happen. While shorter on-times lead to more frequent on-times, they also lead to a larger proportion of on-time energy being spent on boot overhead.

C. OPS Model

Our OPS model takes a similar approach as our TTO model. We discretize time into slots of length s, and consider a given on-time of node A falling in an interval of length T_B. We model OPS as the expected number of overlapping slots in a given time period.

We first find the expected number of overlapping slots for an on-time of node A. This is S, the total number of overlapping slots for all possible rebirth slots of B, divided by the number of slots in T_B. The count for S is complicated by the fact that, in our framework, on-times with overlap are extended until one node or the other runs out of energy. In Fig. 5, we illustrate how we find S, with the following result:

$$S = \sum_{i=b}^{m-1} (K - i) + \sum_{i=b+1}^{n-1} (K - i), \quad (7)$$

where K is the number of slots in the maximum on-time τ_{max}.

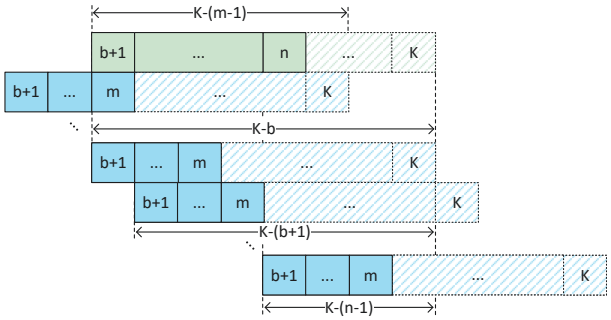


Fig. 5: Counting S, the number of overlapping slots for nodes A (green) and B (blue) for all possible rebirth slots of node B. The overall setup is the same as Fig. 3. For clarity, boot slots are not shown. Here we illustrate a case where m and n are different. The hatched areas indicate the amount that a node can extend its on-time after overlap is detected (up to K, the number of slots in τ_{max}, assumed to be the same for both nodes). In the leftmost rebirth slot for node B that produces overlap, the overlap begins at B's slot m, and the overlap extends for $K-(m-1)$ slots (limited by B's remaining on-time). In the rightmost rebirth slot that produces overlap, the overlap begins at B's slot $b+1$, but here, the overlap is constrained by A's remaining on-time, for $K-(n-1)$ slots of overlap. The alignments between these two extremes can be enumerated using the two midpoints shown, resulting in ranges $[K-b, K-(m-1)]$ and $[K-(b+1), K-(n-1)]$. All other rebirth slots for B produce zero overlapping slots, so summing over these ranges produces S.

To find the expected number of overlapping slots for an on-time of node A, we divide S by the number of slots in T_B. Using Eq. (3), we obtain:

$$E[\text{overlapping slots}] = \frac{S}{\frac{T_B}{s}} = \frac{sP_B S}{P_{\text{ON}} \tau_B}. \quad (8)$$

Finally, to get OPS, we divide this quantity by the lifecycle period of node A. Again using Eq. (3), we obtain:

$$E[\text{OPS}] = \frac{sP_A P_B S}{{P_{\text{ON}}}^2 \tau_A \tau_B}. \quad (9)$$

A contour plot of this quantity, using the same parameters as Fig. 4a, is shown in Fig. 4b. As with the TTO model, the plot is symmetric across the diagonal. Also as before, the lower-left and upper-right corners show the worst performance, and about 5 slots provides the best symmetric on-time. *However, the OPS is maximized when one node chooses $\tau = \tau_{min}$ and the other node chooses an on-time of around 15 slots.* To understand this, we first note that if a node has a small τ and overlap occurs, on average, that overlap will begin earlier in its on-time than for a node with a larger τ. This means the node with a smaller τ can typically extend its on-time further, allowing for more overlapping slots per successful overlap, on average. The point at 15 slots optimally balances the tradeoff between this

phenomenon and the shorter TTO (more frequent on-times) achieved in the lower-right/upper-left corners of Fig. 4a.

D. Discussion

These models are intended to help analyze the design space of our stateless LMP framework. The trends depicted by these models are reflected in reality, as shown in Section VI. The models can also be useful for an order-of-magnitude estimation of TTO and OPS, given input parameters. However, they are not expected to accurately predict absolute values. For example, the T_i used in the models does not take into account the longer off-times experienced by nodes that extend their on-times after overlap. Thus, the model may overestimate performance, particularly for τ values that lead to frequent overlap (though in our target scenario, most on-times do not have overlap, so the effect is small).

By helping to understand the design space, these models can be used to design specific protocols for a variety of scenarios. They could even be used to create an adaptive protocol, where one node dynamically adjusts its on-time to maximize communication opportunities with another node, depending on that node's τ. We leave the design and evaluation of such protocols to future work, and use the remainder of this paper to validate these models.

V. LMP Implementation

In order to evaluate our stateless LMP framework, we prototyped a system capable of performing communication between two batteryless, intermittent nodes solely powered by RF energy harvesting. We describe this system in detail below. We emphasize that this system is built entirely using commercial off-the-shelf hardware.

A. LMP-Capable Batteryless Sensor Node

Our prototype batteryless sensor node, pictured in Figure 6, consists of a main microcontroller (MSP430FR5994 [23]), a radio chip (CC1352R [24]), and an energy-harvesting unit (PowerCast 915 MHz RF energy harvester [25]). The microcontroller supports LMP control and portions of the application that need to be persisted across multiple lifecycle periods.

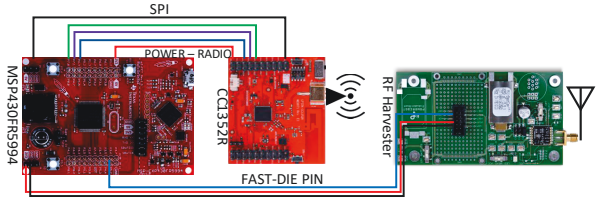


Fig. 6: Diagram of our prototype batteryless sensor node, with an MSP430FR5994 as the main microcontroller and a CC1352R as the radio. The node is powered solely using an RF harvester.

We selected the MSP430 as our main MCU due to its ultra-low power consumption (6.3 mW active and 2.31–3.3 μW standby), fast boot times ($<$ 1 ms from cold boot and $<$ 1.0 μs from standby LPM2/LPM3 modes), and integrated FRAM [26].

Due to the intermittent nature of execution, the main application may require frequent writes to non-volatile memory to persist data and system state between lifecycle periods [27]–[30]. In our prototype setup, we need to store the current LMP configuration and the successfully-transmitted data packets. Writing this frequently to common embedded non-volatile memory, such as NAND flash, is not feasible because of write limitations. For example, in CC1352, the flash is organized as sets of 8 kB blocks, and a 64-bit word can only change the bits from 1's to 0's (a block reset writes 1's). At the same time, only 83 writes can be done within a 256B row before a full sector erase is required (we measured 180 ms to erase a sector – longer than many of our LMP on-times). Any number of writes greater than this may cause programming of unwritten bits in the rows previously erased [31].

While emerging non-volatile technologies like ferroelectric RAM (FRAM), magnetoresistive RAM (MRAM), and resistive RAM (ReRAM) are promising technologies for future intermittent sensor nodes, MSP430 is the only commercially-available MCU that integrates one of these technologies (FRAM). Each byte location in FRAM can be independently updated in one to two MCU cycles without ever erasing any other location. This allows for in situ non-volatile checkpointing of program contents like the stack, the register contents, and other values that are updated during each on-time.

We selected the CC1352R radio chip because it natively supports a wide range of frequencies (169–2480 MHz ISM band with 4 kHz bandwidth) and most of the low-energy, two-way communication protocols across these bands, especially in sub-1 GHz. The MSP430 is connected to the radio chip via SPI to send LMP control commands and data to transmit.

We used the PowerCast 915 MHz energy harvester to harvest and regulate RF power from an RF transmitter. The harvester module charges a capacitor (we used the default 50 mF capacitor in our experiments) to an "on" threshold of 1.23 V, at which point the regulator is enabled. The regulator is disabled once the capacitor voltage reaches 1.02 V again. Critically, the PowerCast harvester also allows the MSP430 to disable the regulator early via a reset pin. This provides the *early-die* capability used as the control mechanism by our stateless LMP framework. Given the capacitor size, voltages, and power consumption, the node's experimental τ_{min} and τ_{max} are 18 ms and 293.5 ms, respectively.[1]Note that the latter value varies with the harvesting power (and therefore, lifecycle period) since the PowerCast regulator uses current from both the harvester and the capacitor to power the MCU and radio.

B. LMP System Software

While LMPs can be implemented bare metal on a sensor node, we chose to develop our prototype on top of an operating system to allow applications to be easily built on top of the LMP+OS implementation. However, the boot time of the OS is a major consideration for an intermittent node, especially

[1] τ_{min} is the minimum observed on-time, including boot-time, when a node is configured to only be on for one slot. τ_{max} is the maximum observed on-time duration of a node when it does not early-die.

one running an LMP that is configured to repeatedly turn on, quickly attempt communication, and turn off again. Table I compares the boot times of two popular OSes for wireless sensor nodes (Contiki OS [32] and TI-RTOS [33]) across three MCUs (MSP430, CC1352, and CC2650 [34])[2]. On the CC2650, we observed that Contiki in its out-of-the-box configuration has a boot time of at least 100 ms—over 5× our typical τ_{min}, even before the radio is initialized. According to our measurements, it takes an additional 50 ms to initialize the radio on Contiki. In contrast, for TI-RTOS, we measured a boot time of less than 28 ms on the CC2650 and below 25 ms on the CC1352, and initializing the radio requires less than 1 ms. TI-RTOS also boots similarly quickly on the MSP430. Therefore, we use TI-RTOS as the OS for both MCUs.

TABLE I: Comparison of OS boot times (in ms) of MSP430, CC1352, and CC2650 devices using TI-RTOS and Contiki.

Device	MSP430	CC1352		CC2650			
OS	TI-RTOS					Contiki	
W/radio	×	×	✓	×	✓	×	✓
Min	3.4	5.7	7.1	12.8	13.2	100.8	147.9
Average	8.5	14.1	14.7	20.5	20.7	107.9	153.2
Max	16.7	23.9	24.9	27.1	27.4	113.7	160.3

In our software, nodes communicate using TI's EasyLink radio driver [35]. The radios are configured to transmit and receive in the sub-1GHz (779–930 MHz) band. The communication consists of a two-way handshake: the sender unicasts a packet to the receiver, and the receiver responds with a software acknowledgement (ACK). For a particular LMP configuration (LMP-X), the sender transmits X packets, waiting for an ACK after each packet transmission. If it does not receive an ACK after X packets, it early-dies (i.e., disables the energy harvester's voltage regulator). On the other hand, if the sender receives an ACK, it continues to communicate until it runs out of energy or times out (i.e., it does not receive an ACK for a packet and thus assumes the receiver has run out of energy). Correspondingly, the receiver running LMP-X listens for $X \times 5$ ms to receive a packet from the sender.[3] As the receiver successfully receives packets, it extends its listening time by 5 ms until it runs out of energy; otherwise it disables its regulator. We also implement a special case we call LMP-Null. In this configuration, instead of counting packets (or receive slots), the node simply remains on until it dies. We consider this as the baseline case, as it is the default behavior of a typical intermittent system that does not use an LMP.

C. Evaluation Setup

To validate our stateless LMP framework, we used two of our prototype intermittent nodes. One node acts as a sender and the other node acts as a receiver. Both sender and receiver harvest energy from a PowerCast 915 MHz RF transmitter [36] and have no other power source. Communication occurs as described in the previous section, using data packets and ACKs with a 4-byte payload.

As shown in Fig. 7, a USB-powered sniffer node (connected via diodes to avoid artificially increasing the on-time of nodes) records the experimental statistics, such as the number of packets transmitted, the number of acknowledgements received, and on-times and off-times of both nodes. The sniffer timestamps these events using an on-board timer and sends them to a host computer via UART for logging and analysis. We target a particular lifecycle ratio by tweaking the distance and orientation between the RF transmitter and each node until the desired lifecycle ratio is observed. The lifecycle ratio varies over time, but we found that it remains relatively stable for the duration of the experiments. When an experiment runs, the nodes automatically (as controlled by the sniffer) sweep through different LMP configurations of both sender and receiver. In an experiment, each lifecycle ratio configuration runs for either 30 minutes when nodes have a relatively high lifecycle ratio or 120 minutes for lower lifecycle ratios where our model predicts significantly larger TTOs (i.e., 20 × longer). Our longest experiments ran for 50 hours total.

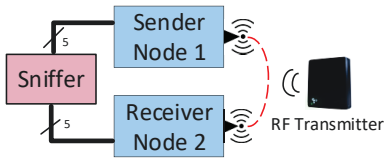


Fig. 7: Evaluation setup for the stateless LMP implementations.

VI. Experimental Evaluation

We evaluate our LMP implementations using two metrics—time-to-handshake (TTH) and throughput. These metrics are analogous to TTO and OPS, described in Section III-B. The difference is that TTH measures the time from the last successful communication in an overlapping on-time until the first successful communication of the next overlapping on-time. Throughput measures the overall amount of data (payload) successfully sent from the sender to the receiver. Successful communications are those where the sending node receives an ACK—therefore, these metrics are conservative, measurable estimates of TTO and OPS (which only consider overlap in an ideal sense). Aside from demonstrating our implementation, the purpose of our experiments is to validate the models presented in Section IV. We reiterate that the goal of the models is not to perfectly predict the actual TTH and throughput of our system, but to correctly predict which LMP configurations should be used to optimize these metrics.

A. LMP Communication Performance

In our first experiment, the sender has an average lifecycle ratio of 13.3% and the receiver has an average lifecycle ratio of 7.6%. Fig. 8 shows the TTO, as predicted by the model, and TTH, as measured by the experiments, at select LMP configurations (LMP-1, -5, -10, -15, and Null). The experimental results clearly follow the trends predicted by the model. The best median TTH (2.85 s) occurs at extreme asymmetry, when

[2]We initially considered using the CC2650 while choosing our OS. The CC1352 boot times for Contiki would likely be similar.

[3]The slot time of 5 ms was chosen by experimentally measuring the round-trip time of a packet sent and ACKed.

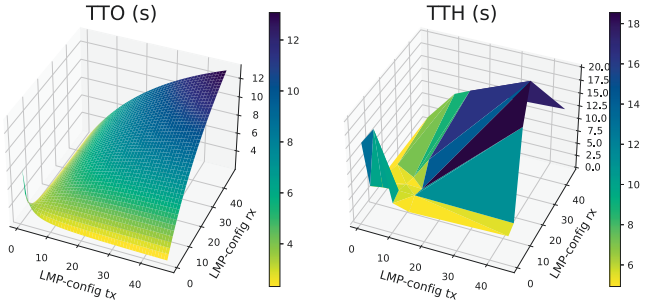


Fig. 8: TTO (model) and TTH (prototype) vs. LMP configurations, with avg. lifecycle ratio of 13.3% for sender and 7.6% for receiver.

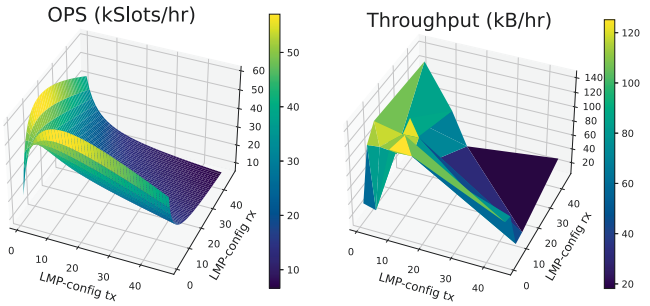


Fig. 9: OPS (model) and throughput (prototype) vs. LMP config, with avg. lifecycle ratio of 13.3% for sender and 7.6% for receiver.

one node uses LMP-1 and the other uses Null-LMP, since this increases the probability on-times overlap. Relatively low TTHs can be achieved when one node is executing Null-LMP, having longer on-times, and the other uses shorter on-time LMPs (LMP-1 to LMP-10). Symmetric LMP-5 and LMP-10 configurations also produce low TTHs.

Figure 9 shows the respective OPS and throughput for the selected LMP configurations with the 13.3% (Tx) and 7.6% (Rx) lifecycle ratios. As our model predicts, the lowest latency LMP combination (1-Null) does not match the highest throughput LMP combination (1-15). This happens because the 1-Null combination has a lower average overlap per on-time, since the node running Null-LMP, on average, has already consumed approximately half of its on-time energy by the time overlap occurs. This behavior, predicted in the model, can be seen in Fig. 9, where the maximum throughput of 146.16 kB/hr is observed when the sender does LMP-15 and the receiver does LMP-1.

B. Lifecycle Ratio and Boot Time Sensitivity Study

Given that our LMPs are capable of communicating at an already low lifecycle ratio, we test a more severe lifecycle scenario, where the receiver has an average lifecycle ratio of 5.55% and the sender only has a lifecycle ratio of 1%. Figure 10 and Fig. 11 show the impact of these decreases when compared to the previous figures. The change in lifecycle ratio directly increases the TTH and reduces the throughput of the system, since the nodes can only be on for a more limited amount of time. In other words, the intermittency imposes tighter constraints on the system. For the experimental

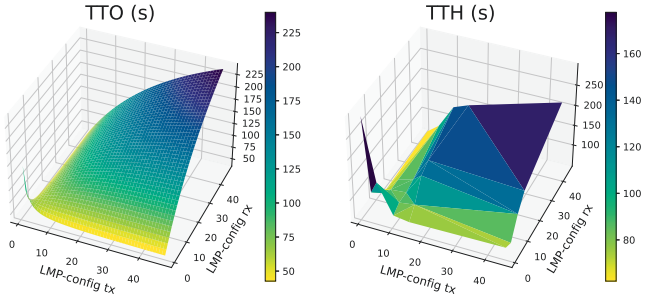


Fig. 10: TTO (model) and TTH (prototype) vs. LMP configurations, with avg. lifecycle ratio of 1% for sender and 5.55% for receiver.

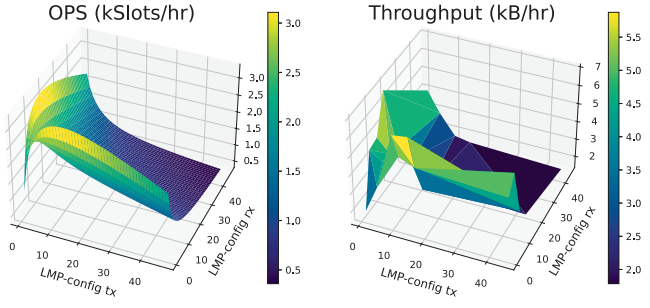


Fig. 11: OPS (model) and throughput (prototype) vs. LMP config, with avg. lifecycle ratio of 1% for sender and 5.55% for receiver.

results, at the optimal LMP configurations (1-Null for TTH and 1-15 for throughput), TTH increased 17.6× and throughput decreased 20.8×. However, crucially, the overall trend for both throughput and TTH remain similar to the model *and to the results with higher lifecycle ratios*. Indeed, these experimental results confirm the finding in Section IV-B that a LMP using our stateless framework does not need to adapt to lifecycle ratios to maintain expected optimal communication performance.

Finally, we investigate how boot time impacts our LMP implementation. Table II shows the effect of artificially increasing the boot time on TTH and throughput when both nodes run LMP-10 and the sender and receiver have lifecycle ratios of 13.3% and 7.6%, respectively. As the boot time increases from 15 ms to 95 ms, TTH increases 17× and throughput decreases 126×, because the node wastes time and energy booting during the relatively frequent on-times, rather than spending that energy attempting to communicate. This has a compounding effect, since that extra wasted energy also must be replaced, increasing the off-time of the node for a given LMP configuration. This degradation in TTH and throughput demonstrates the critical importance of a small boot time for a communication LMP.

TABLE II: TTH and throughput vs. boot time for two nodes, both running LMP-10.

Boot time (ms)	TTH (s)	Throughput (kB/hr)
15	5.33	118.47
35	12.272	42.246
55	20.372	19.92
75	40.65	5.644
95	91.33	0.941

VII. Conclusion and Future Work

In conclusion, we see lifecycling of intermittent nodes as a new, distinct problem, particularly in the realm of communication—and we believe this problem needs to be addressed in order to unlock the potential benefits of intermittent sensor nodes. Our first step is to formulate the problem as a protocol problem, to be solved by lifecycle management protocols. In this work, we began a formal exploration of LMPs by proposing a generic framework for stateless LMPs, then analytically modeling expected performance across the design space of the framework. Finally, we implemented this framework on hardware devices and validated our models in a two-node testbed.

Going forward, we believe this space is rich. Lifecycling is a major challenge, but instead of passively accepting the limitations of lifecycling, LMPs actively try to squeeze as much useful on-time as possible out of the system. Our protocol-based approach abstracts the problem, making it ripe for research within the community. We have identified many specific future directions of our research. We plan to use our proposed framework and models to develop concrete protocols for communication between multiple nodes. With the models presented here, we can design protocols to produce optimal behavior under a variety of different scenarios. Another direction is further exploration of stateful LMPs, including design, modeling, and implementation. At a higher level, we are also interested in integrating the concept of LMPs into an embedded OS, and studying the effect that LMPs have not only on communication and the network stack, but also task scheduling, computation, and sensing.

Acknowledgment

This work was supported in part by the U.S. National Science Foundation under Grants 1730275 and 2008548.

References

[1] L. M. Feeney, C. Rohner, P. Gunningberg, A. Lindgren, and L. Andersson, "How do the dynamics of battery discharge affect sensor lifetime?" in *IEEE WONS 2014*.

[2] J. F. Peters, M. Baumann, B. Zimmermann, J. Braun, and M. Weil, "The environmental impact of li-ion batteries and the role of key parameters–a review," *Renewable and Sustainable Energy Reviews*, vol. 67, pp. 491–506, 2017.

[3] M. L. Wymore, V. Deep, V. Narayanan, H. Duwe, and D. Qiao, "Lifecycle management protocols for batteryless, intermittent sensor nodes," in *IEEE IPCCC 2020*.

[4] M. A. Khan, M. Sharma, and B. R. Prabhu, "A survey of RFID tags," *International Journal of Recent Trends in Engineering*, vol. 1, no. 4, p. 68, 2009.

[5] X. Jia, Q. Feng, T. Fan, and Q. Lei, "RFID technology and its applications in Internet of Things (IoT)," in *IEEE CECNet 2012*.

[6] G. Wang, F. Gao, R. Fan, and C. Tellambura, "Ambient backscatter communication systems: Detection and performance analysis," *IEEE Transactions on Communications*, vol. 64, no. 11, pp. 4836–4846, 2016.

[7] Y. Karimi, A. Athalye, S. R. Das, P. M. Djurić, and M. Stanaćević, "Design of a backscatter-based tag-to-tag system," in *IEEE RFID 2017*.

[8] J. Ryoo, J. Jian, A. Athalye, S. R. Das, and M. Stanaćević, "Design and evaluation of "bttn": A backscattering tag-to-tag network," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2844–2855, 2018.

[9] A. Y. Majid, M. Jansen, G. O. Delgado, K. S. Yildirim, and P. Pawełłzak, "Multi-hop backscatter tag-to-tag networks," in *IEEE INFOCOM 2019*.

[10] Y. Sun, O. Gurewitz, and D. B. Johnson, "RI-MAC: a receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks," in *ACM SenSys 2008*.

[11] A. Dunkels, "The ContikiMAC radio duty cycling protocol," 2011, SICS.

[12] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust mesh networks through autonomously scheduled TSCH," in *ACM SenSys 2015*.

[13] R. Piyare, A. L. Murphy, C. Kiraly, P. Tosato, and D. Brunelli, "Ultra low power wake-up radios: A hardware and networking survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, 2017.

[14] J. Oller, I. Demirkol, J. Casademont, J. Paradells, G. U. Gamm, and L. Reindl, "Has time come to switch from duty-cycled MAC protocols to wake-up radio for wireless sensor networks?" *IEEE/ACM Transactions on Networking*, vol. 24, no. 2, pp. 674–687, 2015.

[15] K. Geissdoerfer and M. Zimmerling, "Bootstrapping battery-free wireless networks: Efficient neighbor discovery and synchronization in the face of intermittency," in *USENIX NSDI 2021*.

[16] Y. Wang, Y. Liu, C. Wang, Z. Li, X. Sheng, H. G. Lee, N. Chang, and H. Yang, "Storage-less and converter-less photovoltaic energy harvesting with maximum power point tracking for internet of things," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 2, pp. 173–186, 2016.

[17] A. Hoseinghorban, M. R. Bahrami, A. Ejlali, and M. A. Abam, "Chance: Capacitor charging management scheme in energy harvesting systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 3, pp. 419–429, 2021.

[18] A. Colin, E. Ruppel, and B. Lucia, "A reconfigurable energy storage architecture for energy-harvesting devices," in *ACM ASPLOS 2018*.

[19] J. de Winkel, C. Delle Donne, K. S. Yildirim, P. Pawełczak, and J. Hester, "Reliable timekeeping for intermittent computing," in *ACM ASPLOS 2020*.

[20] V. Deep, V. Narayanan, M. Wymore, D. Qiao, and H. Duwe, "HARC: A Heterogeneous Array of Redundant Persistent Clocks for Batteryless, Intermittently-Powered Systems," in *IEEE RTSS 2020*.

[21] J. Hester, N. Tobias, A. Rahmati, L. Sitanayah, D. Holcomb, K. Fu, W. P. Burleson, and J. Sorber, "Persistent clocks for batteryless sensing devices," *ACM TECS*, vol. 15, no. 4, pp. 1–28, 2016.

[22] V. Deep, A. Mishra, D. Qiao, and H. Duwe, "Revisiting time remanence clocks for energy harvesting wireless sensor nodes," in *ACM ENSsys 2019*.

[23] *MSP430FR5994 LaunchPad™ Development Kit*, Texas Instruments, Sep. 2019.

[24] *CC1352R SimpleLink™ High-Performance Multi-Band Wireless MCU*, Texas Instruments, Jan. 2018.

[25] *P2110B – 915 MHz RF Powerharvester Receiver*, Powercast, 12 2016.

[26] *MSP430FR58xx, MSP430FR59xx, and MSP430FR6xx Family User's guide*, Texas Instruments, Oct. 2017.

[27] K. Maeng and B. Lucia, "Adaptive dynamic checkpointing for safe efficient intermittent computing," in *USENIX OSDI 2018*.

[28] B. Ransford, J. Sorber, and K. Fu, "Mementos: System support for long-running computation on RFID-scale devices," in *ACM ASPLOS 2011*.

[29] M. Hicks, "Clank: Architectural support for intermittent computation," *ACM SIGARCH Computer Architecture News*, vol. 45, no. 2, pp. 228–240, 2017.

[30] H. Jayakumar, A. Raha, and V. Raghunathan, "QUICKRECALL: A low overhead HW/SW approach for enabling computations across power cycles in transiently powered computers," in *2014 27th International Conference on VLSI Design and 2014 13th International Conference on Embedded Systems*.

[31] *CC13x2, CC26x2 SimpleLink™ Wireless MCU Technical Reference Manual*, Texas Instruments, Aug. 2019.

[32] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki—a lightweight and flexible operating system for tiny networked sensors," in *IEEE LCN 2004*.

[33] Texas Instruments, "TI-RTOS: Real-Time Operating System (RTOS) for Microcontrollers (MCU)," https://www.ti.com/tool/TI-RTOS-MCU.

[34] *CC2650 SimpleLink™ Multistandard Wireless MCU*, Texas Instruments, Feb. 2015.

[35] Texas Instruments, "EasyLink RF API for CC13xx/CC26xx family." 2019.

[36] *TX91501B – 915 MHz Powercaster Transmitter*, Powercast, Oct. 2018.