Lifecycle Management Protocols for Batteryless, Intermittent Sensor Nodes

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

***Abstract*—Nodes in batteryless sensor networks operate intermittently, making tasks such as node-to-node communication and coordinated computation extremely challenging. Adding to this challenge, a node typically has little control over its intermittency. Therefore, in this paper, we introduce a new class of protocols, which we call lifecycle management protocols (LMPs), to better control and manage the intermittency of batteryless nodes. These protocols may be designed and optimized for a particular task; here, we propose and evaluate a set of LMPs designed to enable direct communication between intermittent batteryless sensor nodes with active radios.**

I. Introduction

A longstanding dream of wireless sensor network (WSN) and Internet of Things (IoT) research is ubiquitous deployment: sensors everywhere, sensing everything. However, this presents practical problems for traditional battery-powered sensor nodes. Changing the batteries of thousands of sensor nodes would be expensive, time-consuming, and for some applications (embedding in concrete [1], attaching to wind turbine blades [2], etc.), effectively impossible.

The growing paradigm of batteryless, intermittent sensor systems offers an alternative. These systems are powered *solely* by energy harvested from ambient sources, such as radio frequency (RF), vibrations, thermal, wind, or solar. They do not have long-term energy storage (thus are *batteryless*). Instead, as shown in Fig. 1, batteryless systems charge a temporary energy store such as a capacitor until they have enough energy to turn on and perform tasks such as sensing, computation, or communication. When the system runs out of energy, it "dies"—that is, it shuts off completely, losing all volatile state information. It remains dead until it charges to the *on threshold* again, at which point it *revives*. This on-off behavior is called *intermittent* operation.

Intermittent operation creates a number of practical challenges. For example, in order for two wireless sensor nodes to communicate directly with each other, *their radios must be on at the same time*. This is a traditional challenge of WSN research, in which the radios are regularly turned on and off, or duty-cycled, in order to conserve energy. However, intermittent systems add another level of challenge and complexity to this problem: in order to communicate directly between intermittent nodes, *the nodes themselves, not only the radios, must both be on at the same time*. We have found that a typical intermittent system harvesting from an RF power transmitter (PTX) at a distance of just a few meters can have an on-time/off-time ratio of 1% or less, making randomly-aligned on-times highly unlikely. To make matters worse, for many types of ambient energy harvesting, the off-time is subject to unpredictable and highly variable fluctuations in the harvesting rate.

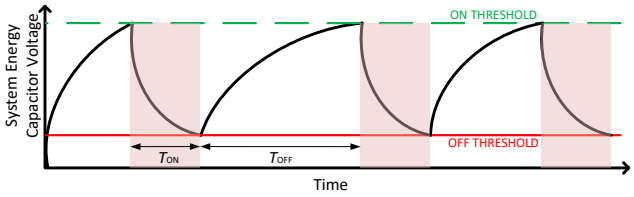

Fig. 1. Illustration of lifecycling behavior of an energy harvesting intermittent sensor node where T_{ON} is the amount of time the node is powered ON and T_{OFF} is the amount of time node is powered OFF. As is typical for intermittent systems, the harvesting rate fluctuates and is too small to sustain continuous operation.

We believe the challenge of controlling the on-times and off-times, or *lifecycle*, of an intermittent node is an interesting and distinct problem. We therefore propose a new class of protocols for this purpose, called *lifecycle management protocols* (LMPs). To the best of our knowledge, this is the first effort to study the control of intermittent nodes' lifecycles as a means of enabling direct interaction between them, such as node-to-node communication. In this paper, we make the following contributions:

- We propose and define the concept of a lifecycle management protocol.
- We propose a set of LMPs designed to enable direct communication between batteryless, intermittent sensor nodes using active radios.
- We evaluate these protocols using simulations based on experimental data acquired from a testbed with Powercast equipment and batteryless, intermittent sensor nodes. We find that using an LMP can improve performance of communication between intermittent nodes by a factor of $10\times$ or more.

The following section defines the concept of an LMP in more detail. In Section III, we present sample LMPs for communication applications, and we evaluate these protocols in Section IV. We discuss related work in Section V.

II. Overview of Lifecycle Management Protocols

We define an *LMP (Lifecycle Management Protocol)* as a protocol for batteryless, intermittent nodes that determines or influences when the node is on or off; that is, it manages the intermittent lifecycle of the node. LMPs control the on-off behavior of the *entire* node, to the extent possible. This differentiates LMPs from any type of traditional protocol, such as duty-cycled wireless MAC protocols, which only turn on and off the *radio* of the node and allow the rest of the node (computation, memory, clocks, etc.) to continue to function. In fact, while we propose LMPs for communication in this work, it is worth noting that the LMP exists outside the network stack, as shown in Fig. 2. In addition to communication, we envision that LMPs may be used to unify management of intermittent operation for a wide variety of purposes, such as optimizing sensor sampling and enabling complex computation routines. Therefore, the LMP potentially interfaces with the user application, the network stack, the core operating system (OS), and even the hardware.

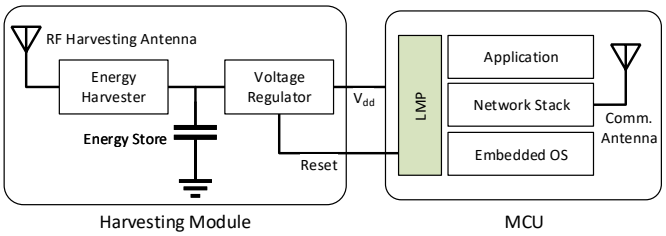

Fig. 2. Example intermittent system architecture showing placement of the LMP. Here we show an RF harvester and LMP control via a Reset pin, but the concept of LMPs can be adapted to other types of harvesting and intermittent system architectures.

A key question for any LMP is: what control is available? In a typical intermittent system, the node has minimal control over its energy supply. In particular, the node cannot control its harvesting rate—this is the primary motivation of proposing LMPs. Assuming a fixed revival threshold, as in the system described in Section I, the node cannot directly control when it revives. It also has no straightforward way to control when it runs out of energy, though it can prolong its life by limiting its operations.

However, our target intermittent nodes have one interesting capability: the ability to "die early." At any point during a node's on-time, the node can manually shut itself off. In our system, this is accomplished via a GPIO pin connected to a *Reset* pin on the harvesting module, as shown in Fig. 2. Dying early means that any remaining energy in the node's energy store is conserved. The node is not fully discharged, so, crucially, it will not take as long to charge back to the revival threshold (variations in harvesting rate notwithstanding).

Using early-die as the control means the LMP is effectively selecting the on-time T_{ON} from some viable range. The off-time T_{OFF} depends on the harvesting rate P_Q and the amount of energy that needs to be harvested to reach the revival threshold. Let P_{ON} denote the average power consumption during on-time. Assuming $P_{\text{ON}} > P_Q$ (as is the case for a typical intermittent system), we can model the off-time in terms of on-time as follows:

$$T_{\text{OFF}} = \frac{T_{\text{ON}}(P_{\text{ON}} - P_Q)}{P_Q}. \tag{1}$$

Then, the ratio of on-time (T_{ON}) to lifecycle length ($T_{\text{ON}} + T_{\text{OFF}}$), which we call L, can be calculated as:

$$L = \frac{T_{\text{ON}}}{T_{\text{OFF}} + T_{\text{ON}}} = \frac{P_Q}{P_{\text{ON}}}. \tag{2}$$

Since P_Q and P_{ON} are beyond the control of the LMP, the ratio L is externally-determined. The LMP cannot control L; instead, it only allows the node to control T_{ON} and the *frequency* at which it experiences on-times.

The early-die capability is the only control lever pulled by the LMPs proposed in this work. It uses existing technology in a commercial off-the-shelf platform. However, we envision that future platforms may have more controls for LMPs to use. These could include hardware capabilities such as an adjustable on-threshold [3] or dynamically-selectable capacitance [4]. Future LMPs could also use software-based techniques, such as integrated lifecycle-informed scheduling of sensing, computation, and communication.

III. Proposed LMPs for Communication

In this section, we present a series of LMPs designed to enable direct communication between intermittent nodes using active wireless radios. An LMP focused on communication should create communication opportunities: times when neighboring intermittent nodes are both on, which we call *overlap*. Since intermittent systems are expected to be low-bandwidth applications, our goal is generally to maximize the number of communication opportunities, i.e., the the frequency of overlap occurrences. We are also interested in shortening the tail of the distribution of time between consecutive overlap occurrences. During overlap, nodes can communicate using existing wireless protocols, such as the low-power MAC and routing protocols designed for wireless sensor networks.

We divide our proposed LMPs into two categories: *stateless* and *stateful*. Stateless LMPs operate with no knowledge of prior events such as communications, whereas stateful LMPs may track communication events and attempt to predict future events. We describe our proposed protocols in the following subsections.

A. Stateless LMPs for Communication

1) Null-LMP: We start with a simple, stateless baseline protocol that we call Null-LMP, shown in Fig. 3. In this protocol, nodes never die early. Once a node turns on, it operates for a length of time $T = T_{\text{max}}$, i.e., until it runs out of energy. Therefore, overlap occurs only by chance. The primary advantage of Null-LMP is that it is simple and may be considered the default behavior of an intermittent node.

Null-LMP makes intuitive and mathematical sense as a baseline. Revisiting Eq. (2), not all of T_{ON} is useful for communication—the node takes some time T_{BOOT} to boot and start up the radio. Considering this overhead, we define a

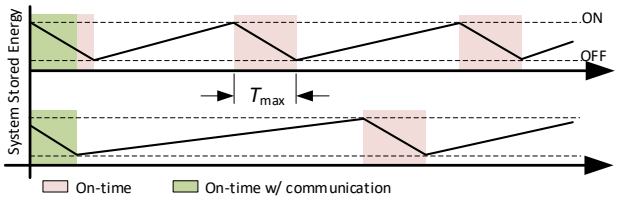

Fig. 3. Null-LMP. Every time a node revives, it remains on for $T_{\max}$. Here the two nodes are initially communicating, but do not overlap again.

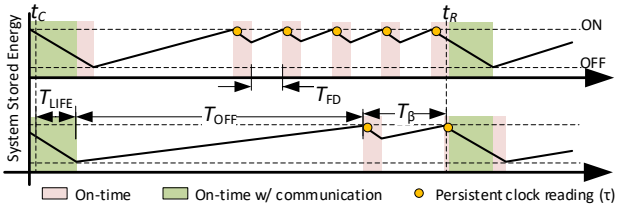

Fig. 5. Clk-LMP. During communication, nodes agree on a rendezvous time t_R, and then use fast-die cycling to try to overlap at this time.

simplified model of the ratio of possible communication time to lifecycle length, denoted as L_C, as:

$$L_C = \frac{T_{\text{ON}} - T_{\text{BOOT}}}{T_{\text{OFF}} + T_{\text{ON}}} = \frac{P_Q}{P_{\text{ON}}} - \frac{T_{\text{BOOT}}}{T_{\text{OFF}} + T_{\text{ON}}}. \quad (3)$$

From this we can see that maximizing T_{ON} maximizes L_C and thus possible communication time per lifecycle, which is exactly what Null-LMP does. However, Null-LMP is not a very effective protocol in most scenarios (as we will show in Section IV). This is because, while Null-LMP maximizes possible communication time for a single node, it does not make any effort to ensure overlap between nodes.

2) Fast-LMP: Our second stateless protocol, called Fast-LMP and shown in Fig. 4, takes the opposite approach of Null-LMP. In Fast-LMP, a node operates for a minimum amount of time T_{min} and, if no communication occurs, it then dies early. If communication does occur, the node remains on until either the communication ends or the node runs out of energy. T_{min} is the minimum amount of time required to establish a communication handshake. We refer to this process of reviving and early-dying after T_{min} as *fast-die cycling*. While Null-LMP maximizes possible on-time, and therefore the single-node probability of overlap at any given revival, Fast-LMP maximizes the frequency of on-times. This means Fast-LMP *attempts* communication many more times (one or two orders of magnitude) in a given time interval. We show in Section IV that, in the end, this increased number of attempts leads to more successful communications in most scenarios, making Fast-LMP more effective than Null-LMP.

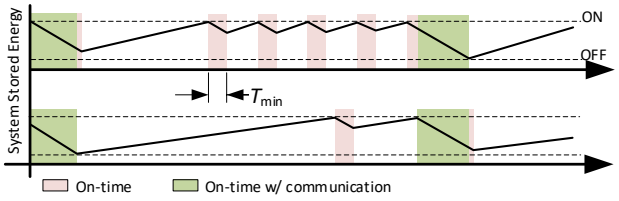

Fig. 4. Fast-LMP. Every time a node revives, it remains on for T_{min}. Here the two nodes are initially communicating, and happen to overlap later in the figure as well.

B. Stateful LMPs for Communication

1) Clk-LMP: While Fast-LMP improves on Null-LMP in practice, it still does not make an explicit effort to arrange overlap between nodes. We therefore propose a stateful version of Fast-LMP called Clk-LMP, which is shown in Fig. 5. In Clk-LMP, during communication, neighboring nodes agree on a *rendezvous* (t_R), a point in time in the future when the nodes will attempt to both be on and communicate again. After charging back to the revival threshold, a node then uses fast-die cycling to "idle" until t_R, at which point it takes a "full" on-time. By increasing the on-time of both nodes around the rendezvous, the protocol attempts to explicitly arrange overlap between the nodes.

Clk-LMP is stateful because it remembers information across off-times. In particular, it needs a *continuous sense of time*, meaning, the ability to track time across both on-times and off-times. The continuous sense of time is used to track the time remaining til t_R. Tracking time during intermittent off-times is non-trivial. Here, we assume the use of a persistent clock [5] to create the continuous sense of time, but other methods for measuring off-times are possible and compatible with Clk-LMP. Briefly, a simple persistent clock functions as follows: during on-time, a known capacitor is charged to a known voltage. During off-time, the charge on the capacitor decays at a known rate. At revival, the voltage on the capacitor is measured and the amount of time that the capacitor spent decaying is estimated using a calibrated mapping table. More advanced persistent clocks may use multiple capacitors to increase accuracy and expand the range of off-times that can be measured [6]. Using a continuous sense of time enabled by a persistent clock, Clk-LMP works as follows.

a) Bootstrapping: Before a node has established communication with any neighbors, the node uses a stateless LMP to bootstrap. Based on the performance of the stateless protocols as presented in Section IV, we choose to bootstrap with Fast-LMP. This means, if there is no agreed-upon t_R, the node simply uses fast-die cycling.

b) Handling Communication Events: When a communication handshake occurs, any planned early-die is cancelled, and the node engages in communication until either the communication session ends (at the discretion of the application or network stack), or the node runs out of energy. During each communication session, the node shares a proposed time-til-rendezvous (I_R) with the other node, calculated as follows:

$$I_R = (T_{\text{LIFE}} + T_{\text{OFF}})(1 + \beta), \quad (4)$$

where T_{LIFE} is an estimate of the remaining on-time for the node, T_{OFF} is an estimate of the upcoming off-time, and β is a safety margin parameter. T_{OFF} and T_{ON} can be dynamically

estimated on the node through a simple mechanism such as a moving average (assuming that values will be similar to the recent past). In our implementation, we use a moving average with a window size of three. T_{LIFE} can be obtained from T_{ON} and the current active clock reading. β is used to improve the chance that the rendezvous will occur *after* the next time both nodes revive by adding a safety margin (shows as $T_\beta = (T_{\text{LIFE}} + T_{\text{OFF}}) \times \beta$ in Fig. 5). This accounts for variations in charging rate and error in predicting T_{ON} and T_{OFF}. In our simulations, we have found that a β of 0.2 provides good performance in most scenarios. After sharing proposed I_R values, the nodes implicitly agree on whichever I_R is larger, and set $t_R = t_C + I_R$, where t_C is the current time at communication.

c) Behavior on Revival: Whenever a node revives, a reading is taken from the persistent clock and τ, the time of revival according the continuous sense of time, is determined. Based on τ's value, one of the following actions is taken:

- If $\tau < t_R - T_{\text{FD}}$, where T_{FD} is an estimate of the off-time of the node during fast-die cycling (obtained e.g. through a moving average), this means that the node can perform at least one more fast-die cycle before the rendezvous, and it does so immediately.
- If $t_R - T_{\text{FD}} \leq \tau \leq t_R$, the node takes a full on-time.
- If $\tau > t_R$, the node has missed the rendezvous. It takes a full on-time in hopes the other node is still on or was also delayed. We select the parameter β to try to avoid this scenario, but it may still occur because of a much slower harvesting rate than expected, for example, due to obstruction of the RF energy signal. In this case, the node targets the "next" rendezvous by setting $t_R = t_R + I_R$.
- Finally, the node reverts back to a bootstrapping protocol (i.e., Fast-LMP) after a chosen number of missed rendezvous attempts (we used three in our evaluations).

The key insight of Clk-LMP is that fast-die cycling can be used to wait for the rendezvous, increasing the probability that the nodes will both be on at t_R. Still, due to variations in the charging rate and imperfections in the continuous sense of time (i.e., clock error), nodes may take their full on-time earlier or later than t_R. However, even if both nodes are not on at t_R, overlap may still occur *close* to the rendezvous with Clk-LMP, because one node is likely to be fast-die cycling when the other takes a full on-time. In this scenario, fast-die cycling guarantees an overlap if:

$$T^A_{\text{FD}} \leq T^B_{\text{max}}, \tag{5}$$

where node A is fast-die cycling while neighbor B takes a full on-time. However, Eq. (5) may not hold in practice, because as a node moves farther from the PTX, T_{FD} grows larger, while T_{max} does not (T_{max} is largely determined by the size of the energy store). We acknowledge, then, that the protocol cannot guarantee a rendezvous in many practical scenarios. In these scenarios, Clk-LMP provides a *probabilistic* guarantee of overlap *near* t_R, with a probability of overlap related to the ratio of T^B_{max} to T^A_{FD}. If overlap does not occur, the nodes independently target the next rendezvous, as discussed above.

2) Clk-LMP-lite: Clk-LMP is tolerant of clock error because nodes attempt a minimal amount of communication during fast-die cycling, providing a reasonable chance of overlap outside of the rendezvous. However, this communication slows down fast-die cycling, increasing T_{FD} and making Eq. (5) harder to satisfy. We therefore propose a variant of Clk-LMP called Clk-LMP-lite that does no communication during fast-die cycling. Upon revival, a node simply checks the time, compares it to t_R, and if not close enough, early-dies without even turning on the radio. This makes Eq. (5) easier to satisfy at a reasonable distance from the PTX. The downside is that overlap can now *only* occur during a full on-time for both nodes. This makes Clk-LMP-lite highly reliant on the arranged rendezvous, and therefore highly reliant on the continuous sense of time. As we will show in Section IV, Clk-LMP-lite can be effective in some scenarios, but only if a highly accurate (i.e. 1% error or less) continuous sense of time is available.

IV. Evaluation

We evaluate the performance of our proposed LMPs in the context of communication using trace-based simulations. In this section, we first describe the simulations and the experimental data used to drive them, and then present results.

A. Experimental Data

We use two types of experimental data as inputs to our simulation: traces of on-time/off-time behavior (*lifecycle data*) of a real batteryless intermittent system, and timing error of a state-of-the-art persistent clock.

1) Lifecycle Data: The lifecycle data consists of on-time/off-time traces of a wireless sensor node powered by an RF energy harvesting system. This data is used to determine on-times and off-times for nodes in the simulation. To gather the traces, we used the setup shown in Fig. 6. The

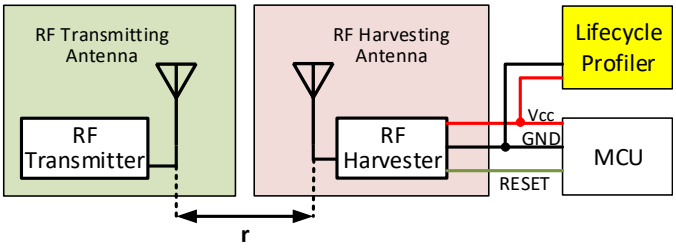

Fig. 6. Experimental setup for measuring the lifecycle of a SensorTag at a distance r from a PTX.

RF power transmitter (PTX) is a Powercast Powercaster [7] that transmits at 915 MHz. This power is harvested by a Powercast Powerharvester [8] at a distance r from the PTX. The harvester stores the harvested energy in a 50 mF capacitor and provides a regulated output to a Texas Instruments (TI) SensorTag [9], which is a wireless sensor node based on the TI CC1352 wireless MCU. The on-time/off-time behavior of the SensorTag is monitored by a lifecycle profiler, which is an MSP430FR5994 launchpad [10] that timestamps the rising and

falling edge of the regulated voltage. The SensorTag can also choose to die early by setting the Reset pin to the harvester. We used this setup to collect lifecycle data at a variety of harvesting distances r. A sample of the lifecycle data gathered is shown in Fig. 7. During the simulation, nodes randomly select on-time and off-time values from this data, based on the distance of the node from the PTX.

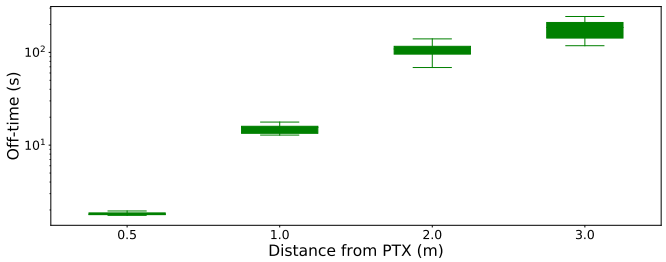

Fig. 7. Box plot of our experimental lifecycle data, which is used to drive the simulations.

2) Persistent Clock Data: To evaluate our stateful LMPs using realistic clock error, we collected clock error data from a state-of-the-art persistent clock. For this experiment, the persistent clock is discharged, and time measurements are read, multiple times for each of a set of known off-times. These measurements are compared to the ground-truth to get a distribution of timing error for each off-time. The simulation randomly selects and applies an error from the corresponding distribution when simulating an off-time measurement. A selection of the clock error data is shown in Fig. 8.

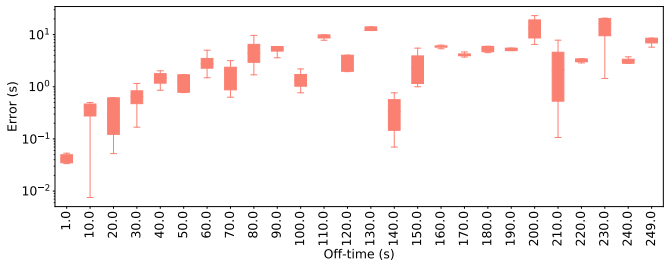

Fig. 8. Box plot of a subset of our experimental persistent clock error data, which is used for the series labeled "w/ PC" in the below figures.

B. Simulator

Our simulator is an event-based simulator written in Python specifically for evaluating the performance of LMPs. For each on-time or off-time of a node, a sample is drawn from the experimental lifecycle data. Charging and discharging of the node is then simulated at each timestep, with the rate calculated from the experimental data sample. To maintain the focus on LMPs, the simulator is designed to detect overlap between nodes, and we assume communication occurs when a defined minimum amount of overlap occurs (8 ms, from empirical observations of the SensorTag). Each simulation begins with a successful communication session. For stateful LMPs, clock error is simulated either by sampling a modeled distribution, or by sampling the distribution of experimental persistent clock data. Table I lists the default simulation parameters—note that many parameters not listed here are encapsulated in the experimental lifecycle data.

We evaluate two metrics: *overlap count*, and *intercommunication delay*. Overlap count is the number of occurrences of overlap that are sufficient for communication in one hour of simulated time, a measure of the number of communication opportunities created by the LMPs. In the below figures, we normalize overlap count to the baseline Null-LMP. Intercommunication delay is the time between consecutive communications in separate occurrences of overlap; to get this result, we simulate until ten separate communication sessions occur. For both metrics, the figures presented below show results aggregated from 100 simulations. Unless otherwise noted, both nodes in the simulation use the lifecycle data gathered 1.0 m away from the PTX.

TABLE I
DEFAULT SIMULATION PARAMETERS.

Parameter	Default Value
Energy store capacitor size	50 mF
Minimum overlap for communication	8 ms
Node bootup time (T_{BOOT})	14 ms
Safety margin parameter (β)	0.2
Node distance from the PTX (r)	1.0 m

C. Results

We present a selection of simulation results evaluating a variety of scenarios. We first calibrate the stateful protocols by evaluating across a range of the β parameter, which determines the size of the safety margin added to all rendezvous times in order to account for varying harvesting rates and off-time estimation errors. In this experiment, the nodes have a perfect continuous sense of time (i.e., no clock error). Results for normalized average overlap count are shown in Fig. 9. We find that $\beta = 0.2$ provides good results for both Clk-LMP and Clk-LMP-lite, so we use this setting in the remainder of the simulations. As expected, when β is very large, Clk-LMP regresses to approximately Fast-LMP, because the nodes spend a long time fast-die cycling before taking a full on-time. Clk-LMP-lite does not communicate during fast-die cycling leading up to a rendezvous, so its number of overlaps approaches zero as β grows. Finally, we see that Fast-LMP has decent performance in terms of overlap count—more than a $5\times$ improvement over Null-LMP in this scenario—primarily due to the large number of overlap attempts it makes.

In Fig. 10, we evaluate the sensitivity of the protocols to clock error, which is key for our stateful LMPs. In this experiment, the clock error is drawn from a modeled distribution. The x-axis shows the expected magnitude of relative clock error. Every time a persistent clock reading is taken in the simulation, an amount of clock error is uniformly selected from the range $[-2x, 2x]$ and applied to the reading. Clk-LMP proves to be robust to clock error, while Clk-LMP-lite does not—above around 3% error, most of Clk-LMP-lite's overlaps

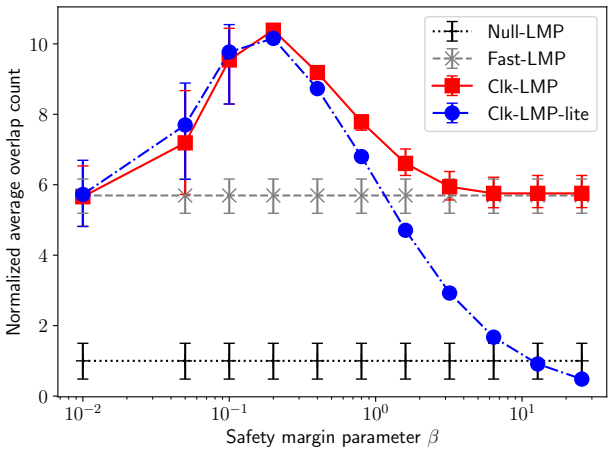

Fig. 9. Normalized average overlap count vs. the safety margin parameter β. Larger is better. The errorbars indicate the 5th and 95th percentiles.

come from falling back to Fast-LMP after multiple failed rendezvous attempts. We also see that Clk-LMP performs better than Clk-LMP-lite in this scenario, even with zero clock error. This is because, with the nodes at 1.0 m, the fast-die off-times are short enough that Clk-LMP has a reasonable probability of overlap even if Eq. (5) does not hold.

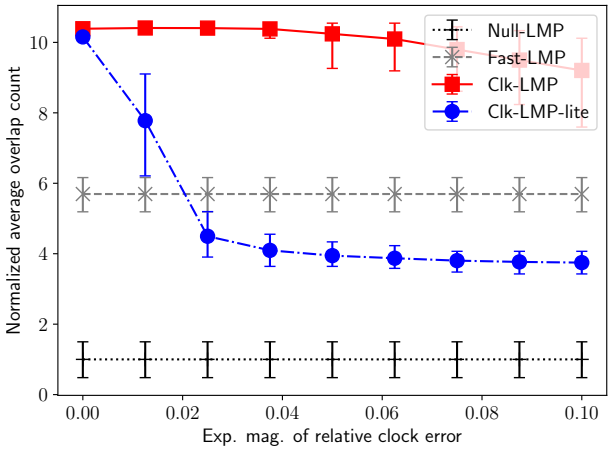

Fig. 10. Normalized average overlap count vs. expected magnitude of the relative clock error, drawn from a uniform distribution centered on zero.

In the remainder of the results, we simulate clock error using our realistic, experimental persistent clock (PC) data. We also evaluate Clk-LMP and Clk-LMP-lite with 1% error, in order to show an optimistic upper bound on the protocols' performance, given a hypothetical highly-accurate continuous sense of time.

Fig. 11 shows normalized overlap count versus distance from the PTX (r in Fig. 6). Here, we fix one node (A) at 1.0 m from the PTX, and vary the distance of node B from the PTX. The first thing to note is that the normalized performance of most protocols increases from 0.5 m to 1.0 m—this is because the baseline, Null-LMP, performs relatively better at 0.5 m, where the ratio of on-time to off-time is high. Matching the observations above, Clk-LMP is insensitive to clock error at 1.0 m. Clk-LMP performs approximately $10\times$ better than the baseline at 1.0 m, even when using the realistic PC error. Clk-LMP-lite w/ 1% also performs well at 1.0 m. At larger distances, we see separation for Clk-LMP based on the amount of clock error, though Clk-LMP w/ PC still performs better than Fast-LMP at all distances. We also see that Clk-LMP-lite performs poorly when using the persistent clock data, with most of its overlap occurring after falling back to Fast-LMP.

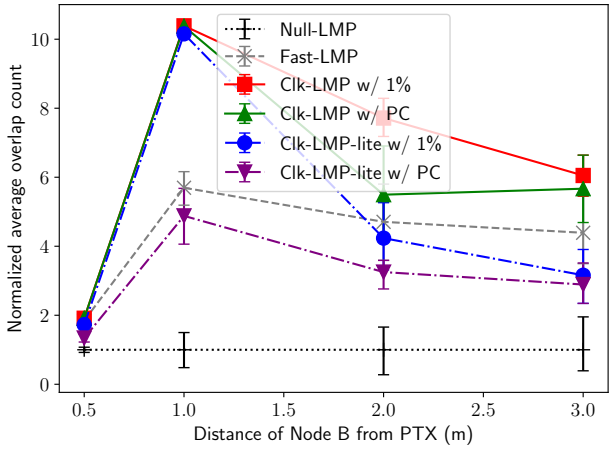

Fig. 11. Normalized average overlap count vs. distance from the PTX. The notation "w/ 1%" means the protocol uses 1% modeled clock error, and "w/ PC" indicates the use of the experimental persistent clock error data.

For an absolute sense of performance, Fig. 12 shows the median and the 95th percentile intercommunication delay versus distance from the PTX (with the y-axis in log scale). Importantly, the stateful protocols (except Clk-LMP-lite w/ PC) achieve a 95th percentile intercommunication delay of around 20 s with both nodes at 1.0 m. Given that full off-times here average around 15 s, this is an impressive result. Another interesting feature is that the performance of the stateful protocols does not change much between 0.5 m and 1.0 m, while the increase in off-times clearly affects the stateless protocols. At larger off-times, the advantage of Clk-LMP begins to diminish as T^A_{FD} from Eq. (5) becomes much larger than T^B_{max}.

Finally, we evaluate the protocols' sensitivity to node boot time (T_{BOOT} from Eq. (3)). Node bootup time affects the ratio between the useful communication time and the total on-time of the node. Our default bootup time is 14 ms, obtained from experimental observations of TI-RTOS running on a CC1352 SensorTag. However, we have also observed significantly longer boot times on other platforms (e.g., >100 ms on Contiki-NG). We therefore present results for normalized overlap count versus node bootup time in Fig. 13. From this figure, we observe that the stateful protocols are more robust to longer bootup times than Fast-LMP. We particularly note that Clk-LMP-lite w/ 1% clock error performs better than Clk-LMP at bootup times longer than 28 ms. A larger T_{BOOT} leads to a larger fast-die off-time, and in this case, Clk-LMP-lite is penalized less because it takes a shorter on-time in the first place (and unlike the distance experiments, the clock

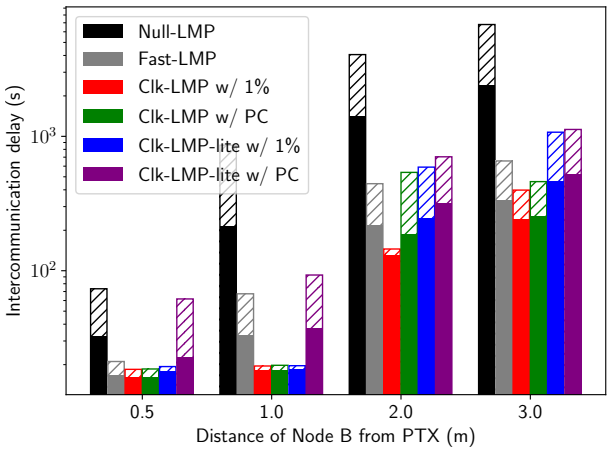

Fig. 12. Intercommunication delay (time between consecutive overlaps) at a variety of distances. Smaller is better. The solid bar shows the median intercommunication delay, while the hatched bar shows the 95th percentile.

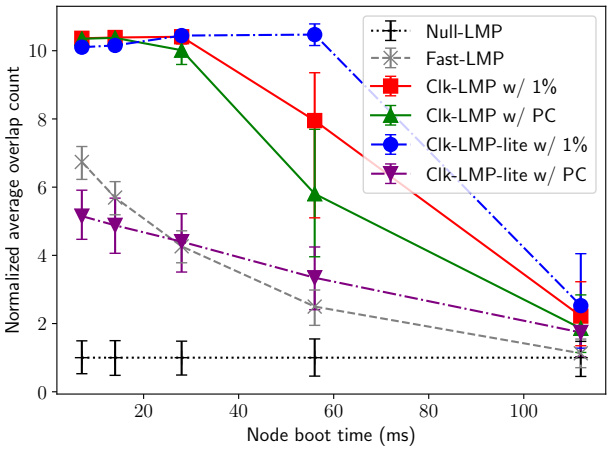

Fig. 13. Normalized average overlap count vs. node bootup time.

errors for full off-times are still small enough for Clk-LMP-lite to be effective). We also see in this figure that Fast-LMP's performance regresses to that of Null-LMP at 112 ms, even though the off-times of Fast-LMP are still not nearly as long as those of Null-LMP. This is the tipping point where Null-LMP's maximization of useful communication time from Eq. (3) begins to outweigh the sheer number of overlap attempts of Fast-LMP.

In summary, we find that our LMPs can improve the communication performance by a factor of $10\times$ or more over the baseline. In particular, Clk-LMP performs well in a variety of scenarios, even with a realistic amount of clock error. Finally, we reiterate that Fast-LMP also performs well for a very simple, stateless protocol, consistently achieving a $5\times$ or more improvement in overlap count compared to Null-LMP.

V. Related Work

A. Intermittent Systems

Works in the emerging area of batteryless, intermittent systems have tackled challenges such as intermittent computing and reliable timekeeping. Intermittent computing [3], [11]–[13] addresses how to perform computational tasks when the system periodically loses all volatile state information. Reliable timekeeping in intermittent systems [5], [6] studies how to maintain a continuous sense of time during off-times, when an active clock is not available.

Another major challenge of intermittent systems is communication. Many intermittent systems adopt a *passive* communication approach, such as passive radio frequency identification (RFID) or other backscatter technology. In these systems, the energy from an incoming radio signal is converted and passively reflected as a response. The response may be received by the signal source (as in an RFID reader) [14], [15], or the response may be intended for a nearby tag, with the signal source potentially far away [16], [17]. The short range of passive communications can be extended with additional energy sources, such as solar [17], [18]. The concept of LMPs may be useful in the context of passive communication, but our work focuses on *active* communication, using an active wireless radio powered by the system's energy store. This approach can achieve longer transmission distances and higher data rates, and it uses mature, off-the-shelf technologies [19]. Additionally, our work here is on communication *directly between intermittent nodes*. Prior work on intermittent node communication has generally focused on the simpler case of single-hop star topologies where the receiver is a constantly-powered node [20], [21].

B. Traditional Low-Power Communication

WSN and IoT research on communication in low-energy scenarios typically assumes a reliable but limited energy source (i.e. a battery). Many *duty-cycled MAC protocols* for WSNs have been proposed. In these protocols, nodes turn their radios off when not in use in order to conserve energy. Approaches can be generally classified as either synchronous (e.g. [22]) or asynchronous, and asynchronous protocols can be further classified as sender-initiated (e.g. [23], [24]) or receiver-initiated (e.g. [25]). An alternative to duty-cycled MAC protocols is the *wake-up radio*, a secondary, ultra-low-power radio that continually listens for a wake-up signal that activates the main radio [26], [27].

LMPs for communication share some concepts with duty-cycled MAC protocols, in that they attempt to arrange overlapping on-time between nodes. Some duty-cycled MAC protocols are even designed to address uncontrolled or cyclical variations in the wireless channel conditions [28], [29], while other work optimizes for the case of energy-harvesting nodes [30], [31]. But duty-cycled MAC protocols attempt to overlap *radio* on-time, which is fully under the control of the node, whereas LMPs attempt to overlap *system* on-time, which is affected by a variety of external factors and is only partially controllable. Furthermore, while LMPs have a limited amount of energy to work with, this energy is more or less free—that is, it recharges in time, and it is wasted if not used. This makes both the design parameters and the design goals of LMPs fundamentally different from duty-cycled MAC protocols. In

the future, duty-cycled MAC protocols, wake-up radios, and WSN techniques for other layers of the network stack may be integrated with LMPs to further improve intermittent node-to-node communication.

VI. Conclusion and Future Work

We believe the concept of lifecycle management protocols (LMPs) is promising for managing and improving the performance of tasks in intermittent nodes. We have presented simulation results showing LMPs providing up to a $10\times$ improvement in communication performance compared to naive baseline behavior. While communication is an obvious application for LMPs, they could also be leveraged to control sensing rates or optimize computations spread across multiple on-times. We are in the process of testing the LMPs presented here on hardware, with promising early results. Future work on LMPs could go in many directions. Future LMPs for communication could dynamically select between stateless and stateful approaches, depending on the environment. We are also interested in exploration of alternative control mechanisms, and extension to multihop communications. Another task for future studies is to design a "whole-system" LMP, i.e. one that takes into account not just communication, but also sensing and computation, and to quantify the overhead introduced by such an LMP. Finally, we note that the results presented here further motivate the exploration of accurate methods of timekeeping during intermittent system off-times.

Acknowledgment

We would like to thank Shen Fu for contributions to the LMP simulator. This work was supported in part by the U.S. National Science Foundation under Grants 1730275 and 2008548.

References

[1] W. Quinn, G. Kelly, and J. Barrett, "Development of an embedded wireless sensing system for the monitoring of concrete," *Structural Health Monitoring*, vol. 11, no. 4, pp. 381–392, 2012.

[2] M. L. Wymore, J. E. Van Dam, H. Ceylan, and D. Qiao, "A survey of health monitoring systems for wind turbines," *Renewable and Sustainable Energy Reviews*, vol. 52, no. C, pp. 976–990, Dec. 2015.

[3] D. Balsamo, A. S. Weddell, A. Das, A. R. Arreola, D. Brunelli, B. M. Al-Hashimi, G. V. Merrett, and L. Benini, "Hibernus++: a self-calibrating and adaptive system for transiently-powered embedded devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 12, pp. 1968–1980, 2016.

[4] A. Colin, E. Ruppel, and B. Lucia, "A reconfigurable energy storage architecture for energy-harvesting devices," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, 2018, pp. 767–781.

[5] 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 Transactions on Embedded Computing Systems (TECS)*, vol. 15, no. 4, pp. 1–28, 2016.

[6] J. de Winkel, C. Delle Donne, K. S. Yildirim, P. Pawełczak, and J. Hester, "Reliable timekeeping for intermittent computing," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 53–67.

[7] *TX91501B 915 MHz Powercaster Transmitter*, Powercast, 10 2018.

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

[9] *CC1350 SimpleLink™ Ultra-Low-Power Dual-Band Wireless MCU*, Texas Instruments, 07 2018.

[10] *MSP430FR5994 LaunchPad™ Development Kit (MSP-EXP430FR5994)*, Texas Instruments, 09 2019.

[11] K. Maeng, A. Colin, and B. Lucia, "Alpaca: intermittent execution without checkpoints," *Proceedings of the ACM on Programming Languages*, vol. 1, no. OOPSLA, pp. 1–30, 2017.

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

[13] A. Y. Majid, C. D. Donne, K. Maeng, A. Colin, K. S. Yildirim, B. Lucia, and P. Pawełczak, "Dynamic task-based intermittent execution for energy-harvesting devices," *ACM Trans. Sen. Netw.*, vol. 16, no. 1, Feb. 2020. [Online]. Available: https://doi.org/10.1145/3360285

[14] A. P. Sample, D. J. Yeager, P. S. Powledge, A. V. Mamishev, and J. R. Smith, "Design of an RFID-based battery-free programmable sensing platform," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 11, pp. 2608–2615, 2008.

[15] K. S. Yıldırım, H. Aantjes, P. Pawełczak, and A. Y. Majid, "On the synchronization of computational RFIDs," *IEEE Transactions on Mobile Computing*, vol. 18, no. 9, pp. 2147–2159, 2018.

[16] V. Liu, A. Parks, V. Talla, S. Gollakota, D. Wetherall, and J. R. Smith, "Ambient backscatter: Wireless communication out of thin air," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 39–50, 2013.

[17] 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 - IEEE Conference on Computer Communications*, 2019, pp. 721–729.

[18] S. N. R. Kantareddy, I. Mathews, R. Bhattacharyya, I. M. Peters, T. Buonassisi, and S. E. Sarma, "Long range battery-less PV-powered RFID tag sensors," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6989–6996, 2019.

[19] C. Delle Donne, K. S. Yildirim, A. Y. Majid, J. Hester, and P. Pawełczak, "Backing out of backscatter for intermittent wireless networks," in *Proceedings of the 6th International Workshop on Energy Harvesting & Energy-Neutral Sensing Systems*, 2018, pp. 38–40.

[20] S. DeBruin, B. Campbell, and P. Dutta, "Monjolo: An energy-harvesting energy meter architecture," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, 2013, pp. 1–14.

[21] A. Sabovic, C. Delgado, D. Subotic, B. Jooris, E. De Poorter, and J. Famaey, "Energy-aware sensing on battery-less LoRaWAN devices with energy harvesting," *Electronics*, vol. 9, no. 6, p. 904, 2020.

[22] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust mesh networks through autonomously scheduled TSCH," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, 2015, pp. 337–350.

[23] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, 2006, pp. 307–320.

[24] A. Dunkels, "The ContikiMAC radio duty cycling protocol," SICS, Tech. Rep. 5128, Jan. 2012.

[25] Y. Sun, O. Gurewitz, and D. B. Johnson, "RI-MAC: a receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks," in *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, 2008, pp. 1–14.

[26] 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 (ToN)*, vol. 24, no. 2, pp. 674–687, 2016.

[27] 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, pp. 2117–2157, 2017.

[28] P. Kindt, H. Jing, N. Peters, and S. Chakraborty, "ExPerio — exploiting periodicity for opportunistic energy-efficient data transmission," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, 2015, pp. 82–90.

[29] M. L. Wymore and D. Qiao, "An opportunistic MAC protocol for energy-efficient wireless communication in a dynamic, cyclical channel," *IEEE Transactions on Green Communications and Networking*, vol. 2, no. 2, pp. 533–544, 2018.

[30] J. Yang and S. Ulukus, "Optimal packet scheduling in an energy harvesting communication system," *IEEE Transactions on Communications*, vol. 60, no. 1, pp. 220–230, 2011.

[31] O. Ozel, K. Tutuncuoglu, S. Ulukus, and A. Yener, "Fundamental limits of energy harvesting communications," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 126–132, 2015.