On the Accuracy of Network Synchronization Using Persistent Hourglass Clocks

Eren Çürük
Ege University
eren.curuk@ege.edu.tr

Kasım Sinan Yıldırım
Ege University and TU Delft
sinan.yildirim@ege.edu.tr

Przemysław Pawełczak
TU Delft
p.pawelczak@tudelft.nl

Josiah Hester
Northwestern University
josiah@northwestern.edu

ABSTRACT

Batteryless sensor nodes compute, sense, and communicate using only energy harvested from the ambient. These devices promise long maintenance free operation in hard to deploy scenarios, making them an attractive alternative to battery-powered wireless sensor networks. However, complications from frequent power failures due to unpredictable ambient energy stand in the way of robust network operation. Unlike continuously-powered systems, intermittently-powered batteryless nodes lose their time upon each reboot, along with all volatile memory, making synchronization and coordination difficult. In this paper, we consider the case where each batteryless sensor is equipped with a hourglass capacitor to estimate the elapsed time between power failures. Contrary to prior work that focused on providing a continuous notion of time for a single batteryless sensor, we consider a network of batteryless sensors and explore how to provide a network-wide, continuous, and synchronous notion of time. First, we build a mathematical model that represents the estimated time between power failures by using hourglass capacitors. This allowed us to simulate the local (and continuous) time of a single batteryless node. Second, we show—through simulations—the effect of hourglass capacitors and in turn the performance degradation of the state of the art synchronization protocol in wireless sensor networks in a network of batteryless devices.

CCS CONCEPTS

• **Networks** → **Network simulations**; • **Computer systems organization** → **Sensor networks**; • **Hardware** → **Power and energy**.

KEYWORDS

Sensor networks, Intermittent computing, Hourglass clocks

ENSsys '19, November 10, 2019, New York, NY, USA

ACM ISBN 978-1-4503-7010-3/19/11…$15.00
https://doi.org/10.1145/3362053.3363497

ACM Reference Format:
Eren Çürük, Kasım Sinan Yıldırım, Przemysław Pawełczak, and Josiah Hester. 2019. On the Accuracy of Network Synchronization Using Persistent Hourglass Clocks. In *ENSsys '19: International Workshop on Energy Harvesting & Energy-Neutral Sensing Systems, November 10, 2019, New York, NY, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3362053.3363497

1 INTRODUCTION

Recent advancements in micro-electronics have enabled harvesting circuits that can efficiently convert and store ambient energy [13]. This led to the emergence of batteryless sensor nodes that can operate by relying on ambient energy sources only [4, 16] which hold promise for replacing existing battery-powered wireless sensor networks. A typical batteryless sensor is composed of several ultra low-power electronic components: a harvesting circuit that targets ambient sources such as solar or radio frequency, an energy reservoir (typically a tiny capacitor), an ultra low-power microcontroller with an integrated non-volatile memory; e.g. TI MSP430 series with FRAM [18], a communication circuitry such as a passive backscatter radio [8] or an ultra low-power active radio [17], and several low-power sensors. When the energy accumulated in the capacitor is above a threshold, the microcontroller and the peripherals start operating to sense the environment, to perform computation on the data and to communicate with other devices. When the energy level drops below the minimum operating voltage, this leads to a power failure and the batteryless sensor dies. When enough energy is accumulated in the capacitor again, the batteryless sensor starts operating. Thus, batteryless sensors operate intermittently.

A power failure resets the volatile state of the batteryless sensor; e.g. the contents of its stack, program counter, and registers are lost. This prevents the progress of computation and makes existing programs and libraries designed for continuously-powered devices useless under intermittent power. There are several efforts that aim to mitigate the effects of power failures to preserve the progress of computation [1, 3, 5, 7, 9–11, 15, 19, 22]. These solutions present runtime libraries that backup the volatile state of the processor into the nonvolatile memory so that the computation can be recovered from where it left-off despite power failures. Moreover, they also ensure that the backed-up state in the nonvolatile memory is always consistent with the volatile state; i.e. they are always equal. It has been shown that several prototype sensing applications can be developed using these runtimes; such as intermittent actuation and activity recognition [22], and greenhouse monitoring [5].

Persistent notions of time: Providing a continuous notion of time is critical to real-time, real world embedded computing applications. Unlike continuously-powered systems, the values of the timer registers in microcontrollers are lost at each power failure and intermittently-powered batteryless devices will lose knowledge of the time upon each reboot. Recent work proposed zero-power timekeepers that use remanence decay to measure the time elapsed between power failures [6, 14]. One approach relies on calculating the percentage of decayed memory cells in an SRAM array to estimate the duration of a power failure. The second approach uses a capacitor as an hourglass: when the device is on the capacitor is charged to a specific voltage, when the device turns off the capacitor slowly begins to dissipate, the elapsed time is estimated by measuring how much the voltage decayed across the capacitor upon reboot. The latter approach is shown to provide a finer grained timing at the cost of additional hardware [6].

The problem statement: In this paper, we consider the case where each batteryless sensor is equipped with a hourglass capacitor to estimate the elapsed time between power failures. Therefore, each batteryless sensor can build a continuous but *local* time notion; namely a *local clock*, by combining its microcontroller's clock together with its hourglass capacitor—the former is more precise and stable but volatile—used when the device is on; the latter is less precise, unstable but persistent despite power failures—used to measure death time. Contrary to the prior work that only focuses on providing a continuous time notion for a single batteryless node, we consider a network of batteryless sensors. We ask: how to provide not only a *continuous* but also a *synchronous* and *global* time notion within this network where each node turns on and off at different times and uses its local clock to measure time intervals.

Contributions: This paper provides two major contributions to the state of the art. Our first contribution is to build a mathematical model that represents the estimated time between power failures by using hourglass capacitors. This allowed us to simulate the local (and continuous) time notion of a single batteryless node. Our second contribution is to show—through simulations—the performance degradation of the state of the art synchronization protocol in wireless sensor networks; namely Flooding Time Synchronization Protocol (FTSP) [12], in a network of batteryless devices where devices uses hourglass clocks to measure time intervals between power failures. Our simulations indicate that even in a small network of ten batteryless nodes, the synchronization performance is approximately 16× worse than a network in which nodes are always on and do not use their hourglass clocks.

2 A MATHEMATICAL MODEL FOR HOURGLASS CLOCKS

Ideally, for a resistor-capacitor circuit with a DC source, the voltage across a capacitor at time $t_0 + \Delta t$ can be represented as

$$V(t_0 + \Delta t) = V(t_0)e^{\frac{-\Delta t}{RC}}, \tag{1}$$

where t_0 is the time at which the capacitor is fully charged and the stored energy starts to dissipate. Therefore, given $V(t_0)$, R and C, the voltage value across the capacitor can be sampled and the elapsed time since the capacitor is fully charged can be calculated as

$$\Delta t = RC \ln\left(\frac{V(t_0)}{V(t_0 + \Delta t)}\right). \tag{2}$$

However, since the sampled voltage readings are subject to measurement errors and also the accuracy of the measurement is dependent on the energy stored in the capacitor, only an estimate of the elapsed time can be obtained. Therefore, instead of the actual Δt, the estimated elapsed time is represented by

$$\hat{\Delta} t = RC \ln\left(\frac{V(t_0)}{\hat{V}(t_0 + \Delta t)}\right), \tag{3}$$

where $\hat{V}(t_0 + \Delta t)$ denotes the inaccurate voltage reading.

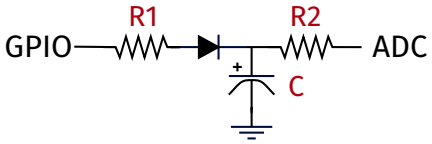

Figure 1: The experimental setup (also depicted in [6, Figure 4]) to sample elapsed time from a hourglass capacitor. We used the following values for the components: R1 = 100 Ω, R2 = 10 KΩ, C = 10 μF, 1n4007 diode. An Arduino evaluation board with 12-bit ADC has been used for driving GPIO input and measuring the voltage across the capacitor.

2.1 Experimental Setup for Data Collection

In order to explore how an hourglass clock behaves in practice, we used the experimental setup depicted in Figure 1—this setup is also presented in [6, Figure 4]. We used an Arduino with 12-bit ADC to sample voltage readings across the capacitor. We programmed the Arduino so that it executed the following loop: (i) the capacitor is charged by driving a GPIO port to high; (ii) the corresponding port is driven to low so that the voltage level across the capacitor starts to decay; (iii) the timer interrupt is generated periodically at every ten milliseconds; and (iv) the voltage level across the capacitor is sampled and logged. The loop (i)–(iv) is executed ten times to collect data from several charge-discharge periods. By using the logged voltage readings from each loop (i)–(iv), we estimated the elapsed time values by using (3).

Minimizing sum of squared errors: We denote the actual elapsed time by $\mathbf{x} = \left[x_1, \ldots, x_N\right]^\mathsf{T}$ and the average of our corresponding estimates by $\mathbf{t} = \left[t_1, \ldots, t_N\right]^\mathsf{T}$. Therefore, we have (x_i, t_i) pairs each capturing the information that states when x_i amount of seconds have passed since the capacitor was fully charged, it is estimated that an average value of t_i seconds have passed by putting the collected voltage measurements in (3). The absolute value of the estimation error of the elapsed time; i.e. $|t_i - x_i|$ is depicted in Figure 2. It can be seen that the accuracy of the estimation is degrading as the voltage across the capacitor is depleted. In order to model this observation analytically, we curve fit to find a representative curve that captures the relationship between $|t_i - x_i|$ and x_i. The curve fitting finds the best curve that minimizes sum of squared errors (SSE)—as depicted by the red curve in Figure 2 (top), which is

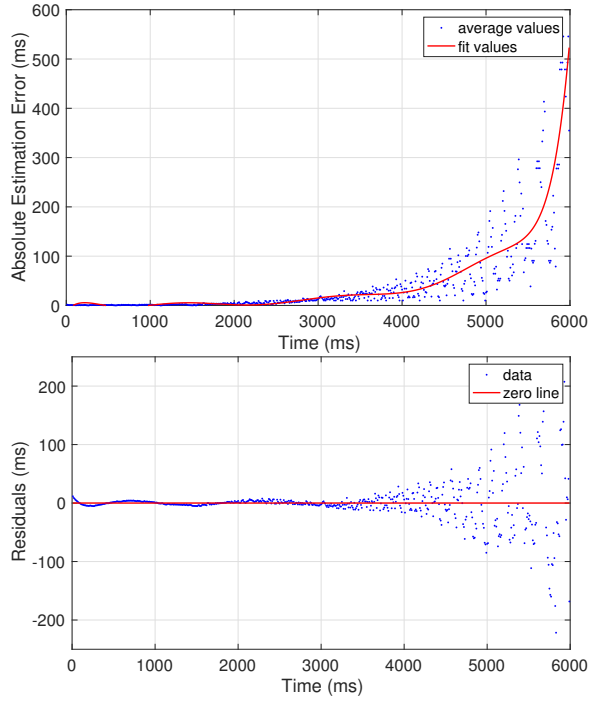

Figure 2: Top of the figure: the Y-axis denotes $|t_i - x_i|$ and the X-axis denotes x_i. The red curve denotes the MATLAB's curve fitting by using a polynomial of order 9—it minimizes sum of squared errors (SSE). Bottom figure: residuals from the curve.

a polynomial of order 9. The residuals depicted in Figure 2 (bottom) also justifies that the accuracy of this model degrades as the voltage across the capacitor is depleted.

Maximizing the likelihood: Instead of curve fitting that considers error minimization, we can also provide a probabilistic perspective to maximize the likelihood of the observed values. To this end, we follow the procedure described in [2]. As indicated previously, the estimated elapsed time by using hourglass capacitors can be represented by the following polynomial curve

$$y(x_i, \mathbf{w}) = w_0 + w_1 x_i + \ldots + w_M x_i^M = \sum_{k=0}^{M} w_k x_i^k \tag{4}$$

where M is the order and $\mathbf{w} = \left[w_0, \ldots, w_M\right]^\mathsf{T}$ are the coefficients of the polynomial. From the residuals depicted in Figure 2, by observation, we can assume that

$$|t_i - y(x_i, \mathbf{w})| \propto x_i^2. \tag{5}$$

and in turn we can model variance of the fitted data by

$$\mathrm{Var}\left[y(x_i, \mathbf{w})\right]) = x_i^4 \sigma^2. \tag{6}$$

Therefore, given the value of x_i, the corresponding value of t_i can be assumed to have a Gaussian distribution where

$$p(t_i|x_i) \sim \mathcal{N}\Big(y(x_i, \mathbf{w}), x_i^4 \sigma^2\Big). \tag{7}$$

Using this model and by assuming that each voltage measurement are taken independently from each other, the likelihood function can be described as

$$\begin{aligned} p(\mathbf{t}|\mathbf{x}) &= \prod_{i=1}^{N} \mathcal{N}\Big(y(x_i, \mathbf{w}), x_i^2 \sigma^2\Big) \\ &= \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi x_i^2 \sigma^2}} \exp\left\{-\frac{(t_i - y(x_i, \mathbf{w}))^2}{2x_i^2 \sigma^2}\right\}. \end{aligned} \tag{8}$$

Therefore, the log-likelihood function is given by

$$\begin{aligned} \ln p(\mathbf{t}|\mathbf{x}) = &- \sum_{i=1}^{N} \frac{(t_i - y(x_i, \mathbf{w}))^2}{2x_i^2 \sigma^2} - \frac{1}{2} \sum_{i=1}^{N} \ln x_i^2 \sigma^2 \\ &- \frac{N}{2} \ln(2\pi). \end{aligned} \tag{9}$$

The value of $\mathbf{w}$ that minimizes the log-likelihood function can be obtained by solving

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^{N} \frac{(t_i - y(x_i, \mathbf{w}))^2}{2x_i^2 \sigma^2}. \tag{10}$$

Using $\mathbf{w}^*$, the σ_*^2 that maximizes the likelihood can be expressed as

$$\sigma_*^2 = \arg\min_{\sigma^2} - \sum_{i=1}^{N} \frac{(t_i - y(x_i, \mathbf{w}^*))^2}{2x_i^2 \sigma^2} - \frac{1}{2} \sum_{i=1}^{N} \ln x_i^2 \sigma^2 \tag{11}$$

which can be found by solving

$$\frac{\partial}{\partial \sigma^2} \left[- \sum_{i=1}^{N} \frac{(t_i - y(x_i, \mathbf{w}^*))^2}{2x_i^2 \sigma^2} - \frac{1}{2} \sum_{i=1}^{N} \ln x_i^2 \sigma^2 \right] = 0. \tag{12}$$

By straightforward algebraic manipulations, we reach

$$\sigma_*^2 = \frac{1}{N} \sum_{i=1}^{N} \frac{(t_i - y(x_i, \mathbf{w}^*))^2}{x_i^2}. \tag{13}$$

2.2 A Probabilistic Model for Hourglass Clocks

In summary, a node can take a voltage sample at time $t = t_0 + \Delta t$ to estimate the actual elapsed time Δt. Based on our previous derivations, this estimate can be represented by $\hat{\Delta} t$ and it can be modeled using the following distribution

$$\hat{\Delta} t \sim \mathcal{N}\Big(y(\Delta t, \mathbf{w}^*), \Delta t^4 \sigma_*^2\Big). \tag{14}$$

Alternatively, we can write

$$\hat{\Delta} t = y(\Delta t, \mathbf{w}^*) + \eta \tag{15}$$

where

$$\eta \sim \mathcal{N}\Big(0, \sigma_\eta^2 = \Delta t^4 \sigma_*^2\Big). \tag{16}$$

3 CONTINUOUS LOCAL CLOCK MODEL

So far, we explored how an hourglass clock can be modeled analytically. In this section, we will provide an analytical model for the *continuous local clock* of a batteryless node by combining the microcontroller's volatile timers together with the hourglass clock. We assume that when a batteryless node has sufficient harvested energy to turn on and operate, it measures the real-time intervals

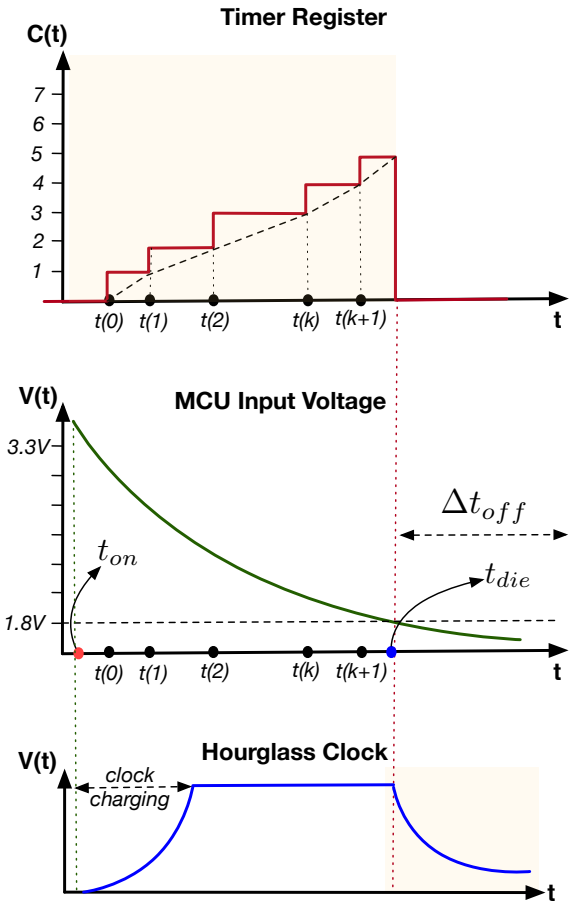

Figure 3: Maintaining the continuous and local clock by combining microcontroller's timer register and hourglass capacitor. When the batteryless sensor has sufficient energy to operate, the microcontroller is active and the timer register can be used as a source of time. When the energy accumulated in the capacitor of the batteryless sensor drops below a predefined threshold, the hourglass clock is used to measure the time between power failures.

by using the timers of its micro-controller. In general, the read-only *timer register* is incremented with each periodical pulse of a crystal oscillator—the oscillator produces an event at time $t(k)$, $k \in \mathbb{N}$.

We follow the notation presented in [21]. Let $C(t)$ denote the timer register, which can be seen as the counter of these events and defined as

$$C(t) = \sum_{k=0}^{\infty} u(t - t(k)) \tag{17}$$

where $u(t)$ is the unit step function defined as

$$u(s) = \begin{cases} 0, & t < 0, \\ 1, & t \geq 0. \end{cases} \tag{18}$$

Table 1: Parameters used in MATLAB simulations

B	f	f_{drift}	W	σ_*^2
10 ms	32 kHz	± 0.1 kHz	8	0.02

In this way the output of the timer register is the step shaped function, as shown in Figure 3 (top). The average oscillator frequency in the interval $\forall t \in [t(k), t(k+1)]$ can be represented by

$$f(t) := \frac{1}{t(k+1) - t(k)}. \tag{19}$$

Using the oscillator frequencies, the timer register $C(t)$ can also be represented by

$$C(t) = \left\lfloor \int_{t_{on}}^{t} f(v)dv \right\rfloor, \tag{20}$$

where t_{on} indicates the time at which the batteryless sensor started operating. It should be noted that upon a power failure, the timer register will reset.

3.1 Combining Timer Register and Hourglass Clock

When the batteryless sensor runs out of energy, it measures the duration between power failures by using the hourglass clock in Figure 3, which is modeled in (15). Using the volatile timer register and the persistent hourglass clock, any node can build a continuous *local clock* that represents the local time, which can be modeled as the following monotonously-increasing function

$$\begin{aligned} L(t) &= L(t_{\text{die}}) + \hat{\Delta}t_{\text{off}} + C(t) \\ &= L(t_{\text{die}}) + y(\Delta t_{\text{off}}, \mathbf{w}^*) + \eta + \left\lfloor \int_{t_{\text{on}}}^{t} f(v)dv \right\rfloor \\ &= L(t_{\text{die}}) + y(\Delta t_{\text{off}}, \mathbf{w}^*) + \eta + (t - t_{\text{on}})\bar{f}, \end{aligned} \tag{21}$$

where $L(t_{\text{die}})$ denotes the value of the *local clock* just before the device dies at time t_{die} due to a power failure, $\hat{\Delta}t_{\text{off}}$ denotes the measured off time, $C(t)$ denotes the time passed since the device starts operating again and $\bar{f}$ denotes the average frequency of the device in the interval $[t_{\text{on}}, t]$. We assume that the local clock $L()$ is maintained in non-volatile memory of the batteryless sensor—$L(t_{\text{die}})$ is saved in non-volatile memory just before the batteryless sensor dies at time t_{die}. Thus, the batteryless sensor node obtains a continuous and local time notion by updating its local clock according to (21). Figure 3 depicts how local clock of the batteryless sensor operates.

4 NETWORK SYNCHRONIZATION USING HOURGLASS CLOCKS

In this section, we consider the problem of how to synchronize the network of batteryless sensors that are equipped with hourglass capacitors as well as timer registers of their microcontrollers. We assume that the batteryless sensors die and wake up at different and arbitrary times and we consider the de-facto time synchronization protocol in wireless sensor networks, namely Flooding Time Synchronization Protocol (FTSP) [12], to synchronize them.

4.1 A Brief Summary of FTSP

In FTSP, a predetermined or dynamically elected reference node r floods its time information L_r periodically at every B seconds into the network. Each direct neighbor v of the reference node stores the synchronization point (L_v, L_r) as a pair (x_i, Y_i) in its least-squares table (LST) of size W that holds the most recently obtained W pairs. It should be noted that upon receiving a new synchronization message, the earliest pair is removed from LST, the received pair is stored in LST. Node v assumes a linear relationship between its local clock L_v and the received local clock L_r of the reference node. It performs least-squares regression by using the stored pairs in order to calculate the estimated local clock value of the reference node. In least-squares regression, the real relationship between L_r and L_v is assumed to be linear; i.e.

$$L_r = \alpha + \beta L_v. \tag{22}$$

By using the pairs $\{(x_i, Y_i) \in LST | i = 0, \ldots, W-1\}$, the estimated slope $\hat{\beta}$ and intercept $\hat{\alpha}$ is calculated as

$$\hat{\beta} = S_{xY}/S_{xx}, \qquad \hat{\alpha} = \overline{Y} - \hat{\beta}\overline{x} \tag{23}$$

where

$$\overline{x} = \sum x_i/W, \qquad \overline{Y} = \sum Y_i/W \tag{24}$$

$$S_{xx} = \sum (x_i - \overline{x})^2, \qquad S_{xY} = \sum (x_i - \overline{x})Y_i. \tag{25}$$

Therefore, batteryless sensor v can estimate the local clock of the reference at any time t by using its local clock L_v and the calculated values $\hat{\alpha}$ and $\hat{\beta}$ as

$$\hat{L}_r(t) = \hat{\alpha} + \hat{\beta} L_v(t). \tag{26}$$

In order to provide network-wide synchronization, batteryless sensor also sends the estimated clock value of the reference node; i.e. $\hat{L}_r$ to its neighboring nodes. Upon receiving this message, the neighboring nodes follow the same procedure to estimate the clock of the reference batteryless sensor. After each batteryless sensor collects sufficient information, the whole network can estimate the clock of the reference node; i.e. the network-wide synchronization is established.

4.2 Simulations

In order to have a first look at the performance of FTSP in a batteryless network, we relied on MATLAB simulations. In order to simulate the local clocks of the batteryless sensor nodes, we used our analytical model described in (21). We used the arbitrarily-selected parameter setting shown in Table 1. We assumed that the microcontroller's timer register operates at 32 kHz with a random drift of ±0.1 kHz, the batteryless nodes communicate with a frequency of 10 ms[1] We did not take into account the message delays occurring due to the communication among the batteryless sensors. We generated arbitrary on and off times for each batteryless node for each interval of 10 ms length by using a uniform distribution.

We performed ten MATLAB simulations where each simulates 500 flooding rounds for FTSP. Figure 4 presents the *average* synchronization errors for the nodes 2, 5, 8 and 10 on a line topology network during these simulations. In order to explore how hourglass clocks and in turn the death time of the nodes effect the synchronization accuracy, we simulated two cases: (i) the nodes die and wake-up arbitrarily and use their hourglass clocks to measure the intervals between power failures as well as microcontroller's timer counter to measure time intervals when they are on; (ii) the nodes are always on and they use their microcontroller's timer counter only to measure time intervals. It can be observed that as the distance to the reference node increases, the synchronization accuracy degrades for both cases. However, the inaccuracy and instability of the hourglass clocks have a multiplicative effect on the synchronization accuracy; in particular for the far-away nodes.

As can also be observed from Figure 5, which depicts the *maximum* synchronization error during our simulations, batteryless node 10 has approximately 16× performance degradation. Our simulations indicate that even in a network of small size in diameter exhibits a poor synchronization performance with hourglass clocks—with a communication frequency of 10 ms, the batteryless node at the end of the line topology had a synchronization error of 35 ms. In the setting where batteryless sensors do not die and operate continuously as in wireless sensor networks, node 10 exhibited a synchronization error of approximately 2 ms.

[1]This can be seen as impractical in a real-world setting but even in this ideal setup, it is possible to get an intuition about the distortion of synchronization due to hourglass clocks.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we considered a batteryless sensor network where each node was equipped with an hourglass capacitor to estimate the elapsed time between power failures. We built a mathematical model that represents the elapsed time and we showed—through simulations—the performance of the state of the art multi-hop synchronization protocol in wireless sensor networks when hourglass capacitors are used to measure time.

Since we are unaware of a real networking setup for batteryless sensors operating intermittently, future work can target the real-world implementation of the proposed ideas and the evaluation by considering actual values rather than simulations. Moreover, other protocols for batteryless sensors, such as proportional-integral controller-based time synchronization [20] can also be implemented by using the hourglass clocks and their performance can be evaluated by using the analytical clock model proposed in this article.

ACKNOWLEDGMENTS

We would like to thank our anonymous reviewers for their constructive criticism. We express our gratitude to Carlo Delle Donne for helpful brainstorming during the project. Dr. Pawełczak is supported by the Netherlands Organisation for Scientific Research, partly funded by the Dutch Ministry of Economic Affairs, under TTW Perspectief program ZERO (P15-06) within Project P4. Dr. Hester is supported by the National Science Foundation under award number CNS-1850496. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

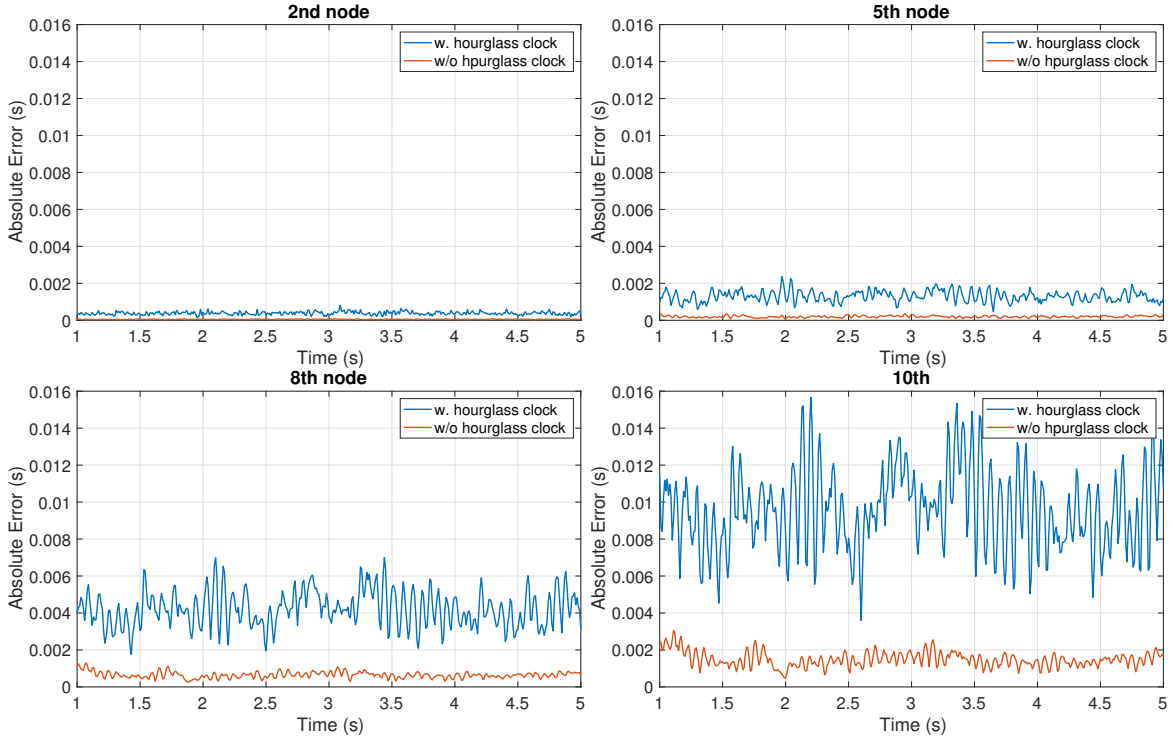

Figure 4: The evolution of the average estimation errors of the nodes 2, 5, 8 and 10 on a line topology network. The figures plot when the nodes are continuously powered and they do not use hourglass clocks (depicted as `w/0 hourglass clock`) and when the nodes are intermittently powered and they use their hourglass clocks to measure time interval between power failures (denoted as `w. hourglass clock`).

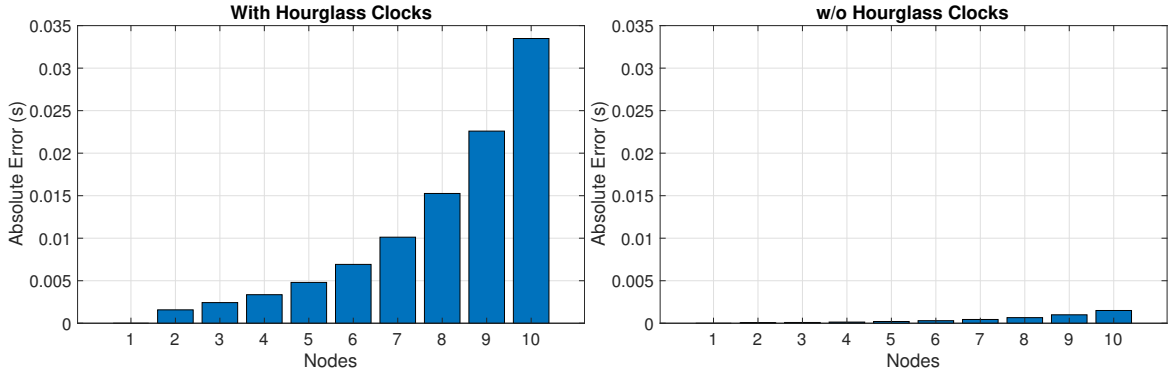

Figure 5: The maximum synchronization error to the reference node (a) when batteryless nodes use their hourglass clocks together with their microcontrollers' timers (left); (b) when batteryless nodes do not die due to power failures and use their microcontrollers' timers only.

REFERENCES

[1] Naveed Bhatti and Luca Mottola. 2017. HarvOS: Efficient Code Instrumentation for Transiently-powered Embedded Devices. In *Proc. IPSN* (April 18–20,). ACM/IEEE, Pittsburgh, PA, USA.

[2] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

[3] Alexei Colin and Brandon Lucia. 2016. Chain: Tasks and Channels for Reliable Intermittent Programs. In *Proc. OOPSLA* (Oct. 30 – Nov. 4). ACM, Amsterdam, Netherlands, 514–530.

[4] Josiah Hester and Jacob Sorber. 2017. Flicker: Rapid Prototyping for the Batteryless Internet-of-Things. In *Proc. SenSys* (Nov. 6–8). ACM, Delft, The Netherlands, 19:1–19:13.

[5] Josiah Hester, Kevin Storer, and Jacob Sorber. 2017. Timely Execution on Intermittently Powered Batteryless Sensors. In *Proc. SenSys* (Nov. 6–8). ACM, Delft, The Netherlands.

[6] Josiah Hester, Nicole Tobias, Amir Rahmati, Lanny Sitanayah, Daniel Holcomb, Kevin Fu, Wayne P. Burleson, and Jacob Sorber. 2016. Persistent Clocks for Batteryless Sensing Devices. *ACM Transactions onn Embeddded Computing*

Systems 15, 4 (Aug. 2016), 77:1–77:28.

[7] Matthew Hicks. 2017. Clank: Architectural Support for Intermittent Computation. In *Proc. ISCA* (June 24–28). ACM, Toronto, ON, Canada, 228–240.

[8] Vincent Liu, Aaron Parks, Vamsi Talla, Shyamnath Gollakota, David Wetherall, and Joshua R. Smith. 2013. Ambient Backscatter: Wireless Communication out of Thin Air. In *Proc. SIGCOMM* (Aug. 12–16,). ACM, Hong Kong, China.

[9] Brandon Lucia and Benjamin Ransford. 2015. A Simpler, Safer Programming and Execution Model for Intermittent Systems. In *Proc. PLDI* (Aug. 13–17). ACM, Portland, OR, USA, 575–585.

[10] Kiwan Maeng, Alexei Colin, and Brandon Lucia. 2017. Alpaca: Intermittent Execution without Checkpoints. In *Proc. OOPSLA* (Oct. 22–27). ACM, Vancouver, BC, Canada, 96:1–96:30.

[11] Kiwan Maeng, Alexei Colin, and Brandon Lucia. 2018. Adaptive Dynamic Checkpointing for Safe Efficient Intermittent Computing. In *Proc. OSDI* (Oct. 8–10). USENIX, Carlsbad, CA, USA.

[12] Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. 2004. The Flooding Time Synchronization Protocol. In *Proc. SenSys* (Nov. 3–5). ACM, Baltimore, MD, USA, 39–49.

[13] Ufuk Muncuk, Kubra Alemdar, Jayesh D. Sarode, and Kaushik Roy Chowdhury. 2018. Multiband Ambient RF Energy Harvesting Circuit Design for Enabling Batteryless Sensors and IoT. *IEEE Internet of Things Journal* 5, 4 (2018), 2700–2714.

[14] Amir Rahmati, Mastooreh Salajegheh, Dan Holcomb, Jacob Sorber, Wayne P. Burleson, and Kevin Fu. 2012. TARDIS: Time and Remanence Decay in SRAM to Implement Secure Protocols on Embedded Devices Without Clocks. In *Proc. Security Symposium* (Aug. 8–10). USENIX, Bellevue, WA, USA, 36–36.

[15] Benjamin Ransford, Jacob Sorber, and Kevin Fu. 2011. Mementos: System Support for Long-running Computation on RFID-scale Devices. In *Proc. ASPLOS* (March 5–11). ACM, Newport Beach, CA, USA.

[16] Alanson P. Sample, Daniel J. Yeager, Pauline S. Powledge, Alexander V. Mamishev, and Joshua R. Smith. 2008. Design of an RFID-based Battery-free Programmable Sensing Platform. *IEEE Transactions on Instrumentation and Measurement* 57, 11 (Nov. 2008), 2608–2615.

[17] Texas Instruments. 2019. CC1101 Low-Power Sub-1 GHz RF Transceiver. http://www.ti.com/lit/ds/symlink/cc1101.pdf. Last accessed: September 2019.

[18] Texas Instruments. 2019. MSP430FR58xx, MSP430FR59xx, MSP430FR68xx, and MSP430FR69xx Family User's Guide. http://www.ti.com/lit/ug/slau367o/slau367o.pdf. Last accessed: September 2019.

[19] Joel Van Der Woude and Matthew Hicks. 2016. Intermittent Computation Without Hardware Support or Programmer Intervention. In *Proc. OSDI* (Nov. 2–4,). ACM, Savannah, GA, USA.

[20] Kasım Sinan Yildirim, Henko Aantjes, Przemysław Pawełczak, and Amjad Yousef Majid. 2018. On the Synchronization of Computational RFIDs. *IEEE Transactions on Mobile Computing* 18, 9 (Sept. 2018), 2147–2159.

[21] Kasım Sinan Yıldırım, Ruggero Carli, and Luca Schenato. 2018. Adaptive Proportional–Integral Clock Synchronization in Wireless Sensor Networks. *IEEE Transactions on Control Systems Technology* 26, 2 (March 2018), 610–623.

[22] Kasım Sinan Yıldırım, Amjad Yousef Majid, Dimitris Patoukas, Koen Schaper, Przemysław Pawełczak, and Josiah Hester. 2018. InK: Reactive kernel for tiny batteryless sensors. In *Proc. SenSys* (Nov. 4–7). ACM, Shenzhen, China, 41–53.