On Distributed Sensor Fusion in Batteryless Intermittent Networks

Kasım Sinan Yıldırım*† and Przemysław Pawełczak†
*Department of Computer Engineering, Ege University, Izmir, Turkey
†Embedded and Networked Systems, Delft University of Technology, Delft, The Netherlands
Email: *sinan.yildirim@ege.edu.tr, †p.pawelczak@tudelft.nl

***Abstract*—In this paper we provide our vision on distributed network composed of backscatter (battery-less) tag-to-tag links. We emphasize that current state-of-the-art literature is focused on centralized computation considering only single-hop tag-to-tag/reader-to-tag links. We observe that collaborative computation is has never been considered before in the battery-less networking. This will bring many advantages for applications (e.g. longer transmission ranges, lower network costs), however introducing new research challenges to the battery-less computation field. To this end, we focus on the well-known distributed sensor fusion problem but in an intermittently-powered sensor network. We show that, even though the batteryless nodes randomly die with a high probability and they neither communicate with their neighboring nodes nor perform computation most of the time, still the simplest implementation of the fully-distributed sensor fusion based on average consensus improves the individual estimations of the batteryless sensor nodes considerably. In the light of this, we anticipate that if the power failure frequencies of batteryless decreases—i.e. the energy is used more efficiently so that they have more opportunity to receive and send packets; sophisticated solutions to handle intermittent communication will not be needed and existing fully-distributed protocols can also be implemented with tiny modifications in intermittently-powered batteryless networks.**

***Index Terms*—Batteryless sensors, Intermittent communication, Distributed sensor fusion**

I. Introduction

Recent advancements in micro-electronics enabled harvesting circuits that can efficiently convert and store ambient energy [1]. This led the emergence of batteryless sensor nodes that can operate by relying on ambient energy sources only [2], [3] and hold promise for replacing existing battery-powered wireless sensor networks. A typical batteryless sensor is composed of several ultra low-power electronic components: an harvesting circuitry that targets ambient sources such as solar or radio frequency waves, an energy reservoir—typically a tiny capacitor, an ultra low-power microcontroller with an integrated non-volatile memory; e.g. TI MSP430FRxx series with FRAM [4], a communication circuitry such as a passive backscatter radio [5] or and ultra low-power active radio [6], 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, compute the perceived the and communicate with other batteryless sensors. When the energy level drops below the minimum operating voltage, this leads to a power failure and the batteryless sensor dies. Therefore, batteryless sensors operate intermittently due to the unpredictable ambient energy and start operating again when there is enough energy accumulated in their capacitors.

A power failure that resets the volatile state of the battery-less sensor; e.g. the contents of its stack, program counter, registers are lost. This prevents the progress of computation and makes existing programs and libraries designed for continuously-powered useless under intermittent power. There are several efforts that aim to mitigate the effects of power failures to preserve the progress of computation [7]–[16]. Roughly speaking, these solutions present runtime libraries that backup the volatile state of the processor into the non-volatile memory so that the computation can be recovered from where it left upon a power failure reboot. 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 [15], and greenhouse monitoring [13].

Problem statement. The communication aspects during intermittent operation on battery-less sensors are overlooked by the researchers. In particular, collaborative and distributed computation can be enabled via networks of battery-less sensors, which require communication capabilities among them. Existing work enabled point-to-point *zero-power* communication using backscatter, e.g. [5], [17]–[19]. However, these works assumed that each batteryless node operates continuously. Therefore, the effect of power failures and in turn intermittent operation on distributed network protocols is overlooked by the community.

Contributions. In this paper, we focus on the well-known distributed sensor fusion problem in networks [34], [35], where the goal is to estimate a parameter collaboratively by considering individual sensor measurements.. However, our focus is a network of intermittently-powered batteryless sensors instead of sensor networks that operate with batteries continuously in the long as well as short term. Our contribution is to show the performance of distributed sensor fusion on an intermittently-powered sensor network via simulations that consider different power failure probabilities. We observed that, even though the nodes randomly die with a high probability and they neither communicate with their neighboring nodes nor perform computation most of the time, still the simplest and naive implementation of the fully-distributed sensor fusion based on average consensus improves the individual estimations of the batteryless sensor nodes considerably. In the light of this observation, we anticipate that by exploiting zero-power communication capabilities; e.g. backscatter communication techniques proposed in [5], [17]–[19], sophisticated solutions to handle intermittent communication can be eliminated by using precious harvested energy effectively to decrease the power failure frequencies of batteryless sensors; so that they have more opportunity to receive and send packets. In other words, we can still exploit the existing fully-distributed protocols that target continuously-operating wireless sensor nodes and apply them with tiny modifications to intermittently-powered batteryless networks.

II. Batteryless Sensing Systems: State of the Art

We classify the state of the art in battery-less sensing systems into two main categories: programming models that enable useful computation on battery-less devices and techniques that enable zero-power communication to transfer information with marginal energy consumption.

A. Batteryless Computation

Since the ambient energy is available intermittently, the energy storage of the battery-less device is depleted frequently—leading to frequent power failures. Therefore, battery-less platforms are by no means useful components of practical sensing and actuation systems unless they satisfy two crucial properties: (i) *forward progress* of computation and (ii) *memory consistency* of the applications despite frequent power failures. Prior works on battery-less computation target ensuring these properties via architectural, compiler and programmer support. There are two main techniques to ensure forward progress and memory consistency: (i) checkpointing and (ii) employing task-based programming models. These techniques create *idempotent* code sections; i.e. sequence of instructions that can be restarted arbitrarily under random power cycles.

Checkpointing-based systems [7], [8], [10] journal the processor's *volatile state*; i.e. registers, stack and heap, in non-volatile memory at specific points in the program code annotated at *compile time* by the programmer or/and the compiler. Upon a system reset after a power failure, the computation is progressed from the latest journaled volatile state for the sake of memory consistency. Other systems [14], [26]–[28] monitor the supply voltage at *runtime* to detect if the voltage is under a pre-defined threshold in order to journal the volatile state. Another system [11] uses an external hardware to ensure memory consistency at run-time and uses a watchdog timer to checkpoint periodically. In task-based systems [9], [12], [13], the programmer decomposes a program into a collection of tasks, provides inter-task input/output relations and control flow at *compile time*. The runtime ensures atomic execution of the tasks; i.e. *all-or-nothing* semantics so that volatile and non-volatile memory is always consistent, and switches to the next task in the control flow. The runtime introduces less overhead as compared to checkpointing-based systems since it journals only the current task pointer and the inputs/outputs of the current task. Unfortunately, all of the aforementioned studies overlook the communication aspects during intermittent operation on battery-less devices.

B. Batteryless Communication

Distributed and collaborative computation requires communication capabilities among the participants so that information can be exchanged. Backscatter enables ultra low-power communication by eliminating energy-hungry hardware component of the active radio and having notably several orders of magnitude less energy requirements than their counterparts [29]. Most of the backscatter networks, e.g. [30], allow communication only between battery-less devices and a dedicated master device. Recent studies addressed this issue and enabled communication among battery-less devices [5], [17]–[19]. In particular, [18], [19] demonstrated a multi-hop backscatter network that can relay information from one physical pount to another. Therefore, we are closer to the fundamental infrastructure for the distributed/collaborative computing: a network of battery-less devices communicating with each other.

However, the effect of the intermittent operation during communication is overlooked by the community. To some extent, data reception and sending operations despite power failures can be handled by using the aforementioned battery-less computation techniques. However, in particular, how to deal with the received data upon recovery from a power failure is an unanswered question. Since almost all sensing applications have temporal requirements, the data exchange during is useful when these requirements are met. On continuous systems, managing the relationship between data and time can be seen as straightforward. On the contrary, time management in battery-less systems is not trivial [13], [31] and explicit checks to consider packet/data expirations is not feasible unless this management is provided.

C. Unified Batteryless Computation and Networking

Considering these fundamental drawbacks, we need to change focus from the centralized computing with a single battery-less entity to the autonomous distributed computation [24], where multiple battery-less tags collaborate in order to reach a common goal, e.g. collaborative sensing, computation and actuation. This fundamental paradigm shift in the intermittently-powered computation domain will bring the following advantages:

Fault tolerance. Tasks, e.g. identical computation blocks, can be handled by multiple entities. Therefore power loss of a single node may not block the computation, since remaining nodes can substitute the failed node and producing the result faster.

Increased service life-time. Since there are multiple entities with different energy levels, the overhead of the computation can be distributed among the tags with more energy, for the sake of the network life-time increase.

Improved time-to-completion. Computationally heavy tasks, such as Fast Fourier Transport, can be distributed among the tags by splitting it into independent blocks. Therefore, parallel execution of the fine grained independent tasks and collecting back the results will improve the overall computation time.

Improved computation accuracy. The concepts of the approximate computing [25] are very relevant in the battery-less computation domain. For instance, software-based techniques, such as loop perforation, some iterations of the loop is skipped to reduce computational overhead but decreasing its accuracy. Distributing the computation among the tags will allow more iterations, and in turn improved accuracy.

Improved sensing accuracy. individual samples from sensors might include errors and the average of the individual sensed values is more robust and trustable. Distributed averaging algorithms and actuation based on the calculated average value become more relevant.

III. Protocol Design Challenges in Intermittently-Powered Networking

In this section we provide an example master/slave distributed protocol for the computation of a workload to provide the challenges of implementing distributed protocols in intermittently-powered networks. The objective the presented protocol is to let many battery-less tags compute the same task so that if one tag dies, others can be able to provide the result—making computation resilient against power failures.

A. Device and Network Assumptions

We envision that a typical battery-less sensor network comprises several tiny sensors equipped with *basic* hardware components *at a minimum* for ultra-low power operation. A backscatter transceiver is required for performing communication without using any power-hungry electronic components. An energy harvesting module transforms ambient energy of radio frequency waves or solar into DC and accumulates it into an energy storage such as a capacitor. A micro-controller (MCU) with persistent memory, e.g. FRAM, performs computation and control operations relying on only the harvested energy. The backscatter transceiver is connected to one of the ports of the MCU, toggling the port to deliver received bits. The MCU is assumed to have several low-power operation modes including a *sleep* mode that consumes marginal current. The MCU can enable/disable the backscatter transceiver by using its another port that powers the backscatter receiver. An external timekeeper with its own dedicated energy store, e.g. [31] is kept alive unless the power-failure is too long—providing a continuous time notion to some extent. An unified battery-less sensor architecture for rapid prototyping battery-less applications is given in [3].

B. An Example Distribute-Collect Protocol

We consider a network composed of a master battery-less node that sends the same piece of information to the selected nodes in the network. The receiver nodes collect information, perform the computation and send back the results to the master node.

Algorithm 1 The pseudo-code for the master tag

```
1: while hasTask do
2:     $w \leftarrow$ GETNEXTTASK
3:     Send $<$ REQUEST, $w >$ to all tags
4:     Receive $<$ ACK, $energy >$ from free tags
5:     Rank tags according to energy levels
6:     Send COMPUTE the first $N$ tags
7:     Collect $<$ RESULT, $r >$ and CHECKPOINT
8: end while
```

Algorithm 2 The pseudo-code for the receiver tags

```
1: ☐ Receive $<$ REQUEST, $w >$
2: Store $w$
3: Send $<$ ACK, $energy >$
4:
5: ☐ Receive COMPUTE
6: Process $w$ and compute $r$
7: Send $<$ RESULT, $r >$
```

The pseudo-code of the distributed-collect protocol for the master node is presented in Algorithm 1. Assume that a master tag has workload denoted by the set of tasks $W = \{t_1, \ldots, t_K\}$. At the beginning of each iteration, the master node picks the next task from this set (Line 2). Then, a computation request is send together with the picked task to the all tags (Line 3) and acknowledgments from the tags are collected (Line 4). The acknowledgments also carry the energy levels of the receiver nodes. The master node ranks the tags according to their energy levels (Line 5) and sends the computation request to N tags that have the highest energy level (Line 6). After the completion of each task, the results are collected is check-pointed so that a after power interrupt the progress will be from the point of failure (Line 7).

The pseudo-code of the distributed-collect protocol for the receiver nodes is presented in Algorithm 2. Upon a request is received from the master tag (Line 1), the received task is stored in non-volatile memory (Line 2) and and the acknowledgment is sent together with the measured energy level (Line 3). Upon a computation request is received (Line 5), the previously-received task w is processed and the result is computed (Line 6). Finally, the result is sent to the master tag (Line 7).

C. Challenges During Intermittent Communication

Algorithm 1 and Algorithm 2 can be interrupted by a power loss at any time—in particular during Send and Receive operations. We will stress how the power interrupts make the design of the distributed protocols more complex. If the master node is interrupted, the iteration is restarted since each at the end of each iteration a checkpoint is performed (Algorithm 1,Line 7). However, the algorithm might be interrupted just before the checkpoint. Consider the case that the master node has already sent requests, received acknowledgments and started to collect the results. It might be the case that some nodes has already sent their results and there is no need to execute the same iteration. Another case is that, the master node can be interrupted just after it sent compute requests (at Algorithm 1,Line 6). It might be the case that after recovery from the power loss, different set of receiver nodes might be selected for the computation. However, the results from the previous set of nodes can be received by the master node. To sum up, power failures during communication introduces several cases that should be considered during the design of the protocols—complicating the design and the implementation.

IV. Fully-Distributed Sensor Fusion in Batteryless Sensor Networks

In this section, we consider the sensor fusion problem in a network of intermittently-powered and distributed sensor nodes. On the contrary to the previous section, we will consider a fully-distributed protocol implementation; rather than a master/slave approach. We will show that fully-distributed protocol requires a tiny modification to its original implementation in wireless sensor networks and it is robust against power failures.

A. Network Model

The network of batteryless sensor nodes is represented by an undirected graph $\mathcal{G}(\mathcal{E}, \mathcal{V})$ where the vertex set $\mathcal{V} = \{1, \ldots, N\}$ includes the batteryless sensor nodes and the edge set $\mathcal{E}$ includes $\{i, j\}$ pairs such that batteryless sensor i can communicate with the batteryless sensor j; and the other way around. Since each batteryless sensor $i \in \mathcal{V} = \{1, \ldots, N\}$ is intermittently-powered and dies/reboots due to power failures, the communication graph $\mathcal{G}$ is not static and time-varying. Therefore, we denote the communication graph at time t by $\mathcal{G}(t) = (\mathcal{E}(t), V)$ such that $\mathcal{E}(t)$ includes the edges among the batteryless sensor nodes that can communicate with each other.

B. Sensor Fusion Background

We first give the fundamental background to perform sensor fusion in a distributed way by summarizing the studies [34], [35]. We assume that a network of batteryless sensor nodes is deployed to estimate an unknown parameter $\theta \in \mathbb{R}^m$; e.g. the location of a moving object [34]. Each batteryless sensor $i \in \mathcal{V}$ accumulates the sensed values $\hat{b}_i \in \mathbb{R}^k$; i.e. the RSSI values, to estimate the parameters θ; e.g. the 3 dimensional axis coordinates of the object. Each sensed value in $\hat{b}_i$ is subject to a measurement error with zero mean. The errors are denoted by the vector v_i and are assumed to be Gaussian:

$$\hat{b}_i = b_i + v_i, \quad \mathbb{E}[v_i] = \mathbf{0}, \quad \mathbb{E}[v_i v_i^\mathsf{T}] = V_i \tag{1}$$

where $V_i \in \mathbb{R}^{k \times k}$ represents the covariance matrix.

Each batteryless sensor has a matrix denoted by $A_i \in \mathbb{R}^{k \times m}$ that relates the unknown parameter vector θ to the measured values b_i. Therefore, for a single batteryless node i the relation between the sensed values and the estimated parameters can be written as:

$$A_i \theta = b_i. \tag{2}$$

However, since the batteryless sensor cannot access b_i and it can only obtain a noisy measurement $\hat{b}_i$, it can only establish the following relationship:

$$A_i \theta = \hat{b}_i. \tag{3}$$

In order to estimate θ, the batteryless sensor i can use the weighted least-squares which represents the maximum likelihood estimate for θ, given by [35], [36]:

$$\hat{\theta}^i_{ML} = (A_i^\mathsf{T} V_i^{-1} A_i)^{-1} A_i^\mathsf{T} V_i^{-1} \hat{b}_i. \tag{4}$$

By using matrix notation, the aggregate measurements of all sensors can be denoted by

$$\mathbf{A}\theta = \hat{\mathbf{b}} \tag{5}$$

where

$$\hat{\mathbf{b}} = \begin{bmatrix} \hat{b}_1 \\ \vdots \\ \hat{b}_N \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} A_1 \\ \vdots \\ A_N \end{bmatrix}, \quad \mathbf{V} = \begin{bmatrix} V_1 & & \\ & \ddots & \\ & & V_N \end{bmatrix} \tag{6}$$

Considering the aggregated measurements, the maximum-likelihood estimate of θ is given by:

$$\hat{\theta}_{ML} = (\mathbf{A}^\mathsf{T} \mathbf{V}^{-1} \mathbf{A})^{-1} \mathbf{A}^\mathsf{T} \mathbf{V}^{-1} \hat{\mathbf{b}}. \tag{7}$$

Using the matrices, the network-wide maximum-likelihood equation can be written as:

$$\hat{\theta}_{ML} = \left(\sum_i A_i^\mathsf{T} V_i^{-1} A_i \right)^{-1} \sum_i A_i^\mathsf{T} V_i^{-1} \hat{b}_i. \tag{8}$$

C. Distributed Average Consensus

In general, if we denote the initial state of the batteryless sensor i by $x_i(0) = y_i$ and denote the set of its neighbors at time t by $\mathcal{N}_i(t)$, the average consensus algorithm can be defined as:

$$x_i(t+1) = W_{ii}(t) x_i(t) + \sum_{j \in \mathcal{N}_i(t)} W_{ij}(t) x_j(t) \tag{9}$$

where

$$W_{ij}(t) = \begin{cases} \frac{1}{N} & \text{if } \{i, j\} \in \mathcal{N}_i(t) \\ 1 - \frac{d_i(t)}{N} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} . \tag{10}$$

Here, $d_i(t)$ denotes the degree of node i at time t. As can be observed, the communication graph is time-varying—represents the intermittently-powered batteryless sensors since they are dying and waking-up due to the depleted energy in their capacitors. It is proven in [34] that if the time-varying communication graphs are **jointly connected** in a *long run*, the average consensus will be reached; i.e.:

$$\lim_{t \to \infty} \mathbf{x}(t) = \left(\frac{1}{N} \mathbf{1}^\mathsf{T} \mathbf{x}(0) \right) \mathbf{1} \tag{11}$$

where $\mathbf{x}(0) = \big[x_1(0), \ldots, x_N(0)\big]^\mathsf{T}$. By jointly connected, it is meant that the union of the set of graphs $\mathcal{G}(0), \ldots, \mathcal{G}(\infty)$ is a connected graph.

D. Distributed Sensor Fusion

In order to calculate Eq. (8) in the distributed setting, an average consensus algorithm is proposed in [34]. As can be observed, the Eq. (8) is composed of two sums. Let us denote these sums by using the following vectors:

$$Y_i(0) = A_i^\mathsf{T} V_i^{-1} A_i \tag{12}$$

$$z_i(0) = A_i^\mathsf{T} V_i^{-1} \hat{b}_i \tag{13}$$

If each batteryless sensor implements the distributed average consensus algorithm Eq. (9) in parallel:

$$\begin{aligned} Y_i(t+1) &= W_{ii}(t) Y_i(t) + \sum_{j \in \mathcal{N}_i(t)} W_{ij}(t) Y_j(t), \\ z_i(t+1) &= W_{ii}(t) z_i(t) + \sum_{j \in \mathcal{N}_i(t)} W_{ij}(t) z_j(t) \end{aligned} \tag{14}$$

then the asymptotic convergence will hold if the communication graphs that occur infinitely often are jointly connected [34]:

$$\lim_{t \to \infty} Y_i(t) = 1/n \sum_i A_i^\mathsf{T} V_i^{-1} A_i, \tag{15}$$

$$\lim_{t \to \infty} z_i(t) = 1/n \sum_i A_i^\mathsf{T} V_i^{-1} \hat{b}_i. \tag{16}$$

As a consequence, each node can obtain the maximum-likelihood estimate of $\hat{\theta}_{ML}$ via:

$$\hat{\theta}^i_{ML} = \lim_{t \to \infty} Y_i(t)^{-1} z_i(t) = \hat{\theta}_{ML}. \tag{17}$$

Algorithm 3 The pseudo-code for the batteryless node i

```
Variables:
    $< Y_i, z_i >$ is kept in non-volatile memory

1: □ Reboot
2: Set a periodic timer
3:
4: □ Reception of $< Y_j, z_j >$ from $j \in N_i$
5: Store $< Y_j, z_j >$ in non-volatile memory
6:
7: □ Timer Fired
8: Update $Y_i$ and $z_i$ via Eq. (14) using stored values
9: Broadcast $< Y_i, z_i >$
```

E. The Distributed Sensor Fusion Algorithm for a Single Node

The algorithm that should be executed by a single batteryless node is depicted in Algorithm 3. In order to implement the algorithm depicted in Eq. (14), there are several crucial points to consider. First, the matrices $Y_i(t)$ and $z_i(t)$ should be saved in non-volatile memory so that the batteryless sensor does not loose its state upon power failures. Second, after a power failure and the node starts operating again, a periodic timer should be set to trigger transmission of these matrices to the neighboring nodes (Lines 1–2). Upon receiving $Y_j(t)$ and $z_j(t)$ from $\mathcal{N}_i(t)$, these values should also be accumulated in a buffer so that the batteryless node considers them when updating its matrices $Y_i(t)$ and $z_i(t)$. It should be noted that these values should also be kept in non-volatile memory so that they are not lost upon a power failure and they will be considered in Eq. (14). Last, when the timer fires, matrices $Y_i(t)$ and $z_i(t)$ is updated and these updated values; the state of node i, is sent to the neighboring nodes. It is worth mentioning that task-based programming models; e.g. [15] or existing checkpoint-based systems; e.g. [16], can be exploited to ensure memory consistency during Lines 4–5 and Line 8 of Algorithm 3.

TABLE I
PARAMETERS USED IN MATLAB SIMULATIONS

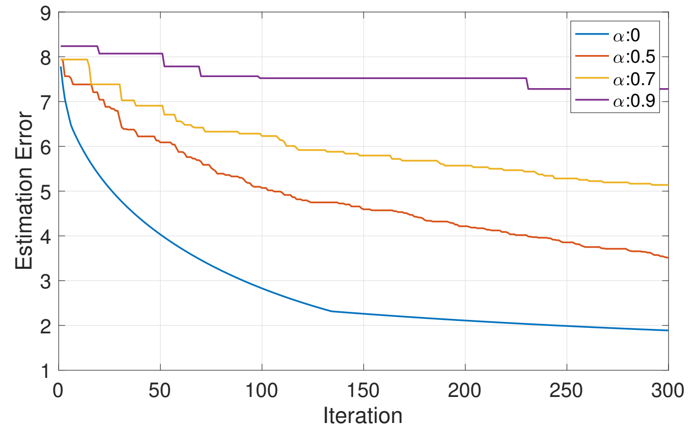
N	α	m	k	iterations
256	[0 0.5 0.7 0.9]	5 10	8	300

V. NUMERICAL SIMULATIONS

In MATLAB, we generated a line topology and a grid topology networks of N nodes. In order to simulate the intermittent execution of the batteryless sensors, we randomly selected the nodes that are operating at each round of the consensus algorithm—we generated random numbers from a uniform distribution and deleted the node if the generated random number is smaller than a predefined failure probability α. Therefore, at each round of the consensus, we assumed that each node is alive with a probability $1 - \alpha$ during the whole consensus round. This randomization is quite reasonable since batteryless sensors are deployed on the same geographical area and it is assumed that they have more or less the same energy harvesting patterns.

The parameter values we used during our simulations are depicted in Table I. In our simulations, we did not consider packet losss during communication. We assumed that nodes implement a Carrier Sense Multiple Access (CSMA) protocol to prevent interference and packet collisions. Therefore, the nodes that are alive during one consensus round successfully communicate with their neighboring nodes; i.e. each node sends its state and receives the states of its alive neighbors. From Fig. 1, it is obvious that the convergence on the grid topology is faster and the steady-state error at any time instant is smaller [34], [35]—this is due to the fact that each node has 4 neighbors in the grid whereas it is only 2 on the line.

When the failure probabilities are considered; even with greater α values the convergence is established but slowly: even a failure probability of 70% (when $\alpha = 0.7$) leads to the convergence so that batteryless sensors estimate the parameter θ collaboratively. This is an interesting result that indicates even though the nodes are operating intermittently, if almost 30% of the nodes are available during a time span that allows them to send and receive data, the sensor



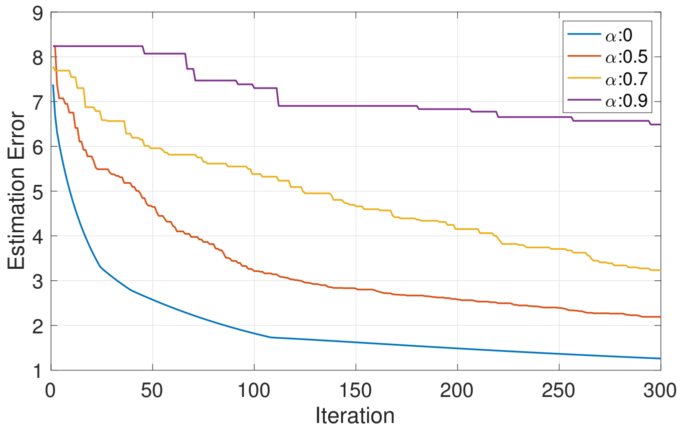


Fig. 1. The maximum error on line (left) and grid (right) topologies with different α values. The estimation error at each iteration is defined as $\max\{\|\hat{\theta}^i_{ML} - \theta\| - \|\hat{\theta}_{ML} - \theta\|\}$.

fusion can be implemented successfully. In other words, the batteryless network can collaboratively improve the estimation of each single batteryless node by implementing the existing distributed sensor fusion algorithm with tiny modifications. This means that, if the availability of the batteryless sensors is increased; i.e. their energy is consumed efficiently to allow them communicate and send data more frequently and in a robust way, the existing fully-distributed protocols designed for continuously-powered wireless sensor networks can still be used in intermittently-powered networks. Our conclusion is that, fully-distributed protocols are robust against power failures and they are more suitable for intermittently-powered sensing systems.

VI. Conclusions

This short paper shed a light on a new, never before considered concept of collaborative computation with the battery-less networks. We focused on the fully-distributed sensor fusion problem and we showed that the simplest and naive implementation of the fully-distributed sensor fusion still improves the individual estimations of the batteryless sensor nodes considerably. In the light of this, we anticipate that if the energy is used more efficiently so that batteryless sensors have more opportunity to receive and send packets, sophisticated implementations of the distributed networking protocols will not be needed and existing fully-distributed protocols can also be implemented with tiny modifications in intermittently-powered batteryless networks. Future work in this domain should address: (i) more efficient battery-less tag-to-tag network hardware implementation and demonstration, (ii) the design and implementation of other fully-distributed protocols for the network of battery-less tags, and (iii) new applications exploiting distributed, battery-less computation.

References

[1] U. Muncuk, K. Alemdar, J. D. Sarode, and K. R. Chowdhury, "Multi-band ambient rf energy harvesting circuit design for enabling batteryless sensors and iot," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2700–2714, 2018.

[2] 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.

[3] J. Hester and J. Sorber, "Flicker: Rapid prototyping for the batteryless internet-of-things," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. ACM, 2017, p. 19.

[4] Texas Instruments, "MSP430FR58xx, MSP430FR59xx, MSP430FR68xx, and MSP430FR69xx Family User's Guide," *SLAU367O*, 2014.

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

[6] Texas Instruments, "CC1101 low-power sub-1 GHz RF transceiver," *SWRS061*, 2017.

[7] B. Ransford, J. Sorber, and K. Fu, "Mementos: System support for long-running computation on rfid-scale devices," *Acm Sigplan Notices*, vol. 47, no. 4, pp. 159–170, 2012.

[8] B. Lucia and B. Ransford, "A simpler, safer programming and execution model for intermittent systems," *ACM SIGPLAN Notices*, vol. 50, no. 6, pp. 575–585, 2015.

[9] A. Colin and B. Lucia, "Chain: Tasks and channels for reliable intermittent programs," in *Proc. OOPSLA*. Amsterdam, Netherlands: ACM, Oct. 30 – Nov. 4, 2016, pp. 514–530.

[10] J. Van Der Woude and M. Hicks, "Intermittent computation without hardware support or programmer intervention," in *Proc. OSDI*. Savannah, GA, USA: ACM, Nov. 2–4, 2016, pp. 17–32.

[11] M. Hicks, "Clank: Architectural support for intermittent computation," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2017, pp. 228–240.

[12] K. Maeng, A. Colin, and B. Lucia, "Alpaca: Intermittent execution without checkpoints," *Proceedings of the ACM on Programming Languages*, vol. 1, no. OOPSLA, p. 96, 2017.

[13] J. Hester, K. Storer, and J. Sorber, "Timely execution on intermittently powered batteryless sensors," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. ACM, 2017, p. 17.

[14] N. Bhatti and L. Mottola, "HarvOS: Efficient code instrumentation for transiently-powered embedded devices," in *Proc. IPSN*. Pittsburgh, PA, USA: ACM/IEEE, Apr. 18–21, 2017.

[15] K. S. Yıldırım, A. Y. Majid, D. Patoukas, K. Schaper, P. Pawelczak, and J. Hester, "Ink: Reactive kernel for tiny batteryless sensors," in *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2018, pp. 41–53.

[16] K. Maeng and B. Lucia, "Adaptive dynamic checkpointing for safe efficient intermittent computing," in *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 2018, pp. 129–144.

[17] 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*, 2018.

[18] J. Ryoo, Y. Karimi, A. Athalye, M. Stanaćević, S. R. Das, and P. Djurić, “Barnet: Towards activity recognition using passive backscattering tag-to-tag network,” in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2018, pp. 414–427.

[19] A. Y. Majid, M. Jansen, G. O. Delgado, K. S. Yıldırım, and P. Pawełczak, “Multi-hop backscatter tag-to-tag networks,” *arXiv preprint arXiv:1901.10274*, 2019.

[20] H. Gao, M. K. Matters-Kammerer, P. Harpe, D. Milosevic, A. van Roermund, J.-P. Linnartz, and P. G. Baltus, “A 60-GHz energy harvesting module with on-chip antenna and switch for co-integration with ULP radios in 65-nm CMOS with fully wireless mm-wave power transfer measurement,” in *Proc. ISCAS*. Melbourne, Australia: IEEE, Jun. 1–5, 2014, pp. 1640–1643.

[21] R. V. Prasad, S. Devasenapathy, V. S. Rao, and J. Vazifehdan, “Reincarnation in the ambiance: Devices and networks with energy harvesting,” vol. 11, no. 1, pp. 195–213, First Quarter 2014.

[22] J. R. Smith, *Wirelessly Powered Sensor Networks and Computational RFID*. New York, NY, USA: Springer Verlag, 2013.

[23] S. Gollakota, M. Reynolds, J. Smith, and D. Wetherall, “The emergence of RF-powered computing,” *Computer*, vol. 47, no. 1, pp. 32–39, Jan. 2014.

[24] D. Peleg, *Distributed computing: a locality-sensitive approach*. SIAM, 2000.

[25] S. Mittal, “A survey of techniques for approximate computing,” *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, p. 62, 2016.

[26] H. Jayakumar, A. Raha, W. S. Lee, and V. Raghunathan, “Quickrecall: A HW/SW approach for computing across power cycles in transiently powered computers,” *ACM J. Emerg. Technol. Comput. Syst.*, vol. 12, no. 1, pp. 8:1–8:19, Jul. 2015.

[27] D. Balsamo, A. S. Weddell, G. V. Merrett, B. M. Al-Hashimi, D. Brunelli, and L. Benini, “Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems,” vol. 7, no. 1, pp. 15–18, Mar. 2015.

[28] 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,” vol. 35, no. 12, pp. 1968–1980, 2016.

[29] P. Zhang, M. Rostami, P. Hu, and D. Ganesan, “Enabling practical backscatter communication for on-body sensors,” in *Proceedings of the 2016 ACM SIGCOMM Conference*. ACM, 2016, pp. 370–383.

[30] P. N. Alevizos, K. Tountas, and A. Bletsas, “Multistatic scatter radio sensor networks for extended coverage,” *IEEE Transactions on Wireless Communications*, 2018.

[31] 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, p. 77, 2016.

[32] V. Liu, A. Parks, V. Talla, S. Gollakota, D. Wetherall, and J. R. Smith, “Ambient backscatter: Wireless communication out of thin air,” in *Proc. SIGCOMM*. Hong Kong, China: ACM, Aug. 12–16, 2013.

[33] K. S. Yıldırım, H. Aantjes, A. Y. Majid, and P. Pawełczak, “On the synchronization of intermittently powered wireless embedded systems,” *arXiv preprint arXiv:1606.01719*, 2016.

[34] L. Xiao, S. Boyd, and S. Lall, “A scheme for robust distributed sensor fusion based on average consensus,” in *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.* IEEE, 2005, pp. 63–70.

[35] F. Bullo, *Lectures on Network Systems*, 1st ed. CreateSpace, 2018, with contributions by J. Cortes, F. Dorfler, and S. Martinez. [Online]. Available: http://motion.me.ucsb.edu/book-lns

[36] G. Strang, *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 2016. [Online]. Available: https://books.google.com.tr/books?id=efbxjwEACAAJ