POLITECNICO DI TORINO
Repository ISTITUZIONALE

Low-power distributed sparse recovery testbed on wireless sensor networks

(Article begins on next page)

25 September 2024

Low-power distributed sparse recovery testbed on wireless sensor networks

Riccardo R. De Lucia, Sophie M. Fosson, Enrico Magli

Department of Electronics and Telecommunications (DET), Politecnico di Torino, Italy

name.surname@polito.it

Abstract—Recently, distributed algorithms have been proposed for the recovery of sparse signals in networked systems, e.g. wireless sensor networks. Such algorithms allow large networks to operate autonomously without the need of a fusion center, and are very appealing for smart sensing problems employing low-power devices. They exploit local communications, where each node of the network updates its estimates of the sensed signal also based on the correlated information received from neighboring nodes. In the literature, theoretical results and numerical simulations have been presented to prove convergence of such methods to accurate estimates. Their implementation, however, raises some concerns in terms of power consumption due to iterative inter-node communications, data storage, computation capabilities, global synchronization, and faulty communications. On the other hand, despite these potential issues, practical implementations on real sensor networks have not been demonstrated yet. In this paper we fill this gap and describe a successful implementation of a class of randomized, distributed algorithms on a real low-power wireless sensor network testbed with very scarce computational capabilities. We consider a distributed compressed sensing problem and we show how to cope with the issues mentioned above. Our tests on synthetic and real signals show that distributed compressed sensing can successfully operate in a real-world environment.

I. Introduction

The recovery of sparse signals in distributed systems has become an important topic in the last few years [1]. On one hand, sparsity has attracted interest for its ubiquity in physical problems (many signals have sparse representations), and due to the advent of compressed sensing (CS) theory [2], [3], [4], which states the conditions to recover sparse signals from compressed measurements. On the other hand, distributed systems, such as wireless sensor networks (WSNs), have become very popular to deal with the necessity of extensive data acquisition and monitoring. Merging sparse signal recovery and networks has given rise to distributed compressed sensing (DCS), which has the goal of increasing networks efficiency through CS.

Early DCS models [5], [6], [7] exploited CS to minimize communications towards a fusion center (FC). Specifically, they envisaged a network in which each node acquires and compresses data according to the CS paradigm; after that, each node transmits such compressed data to the FC, which performs (centralized) sparse signal recovery. More recent works [8], [9], [10], [11], [12], [13], instead, focus on the fully distributed case where no FC is used: both acquisition and recovery are performed in-network, leveraging on local cooperation between nodes. This is a good choice when a FC is not available or far to reach. For example, one can think of WSNs deployed on the territory for monitoring purposes, which collect (compressed) data, and at fixed time intervals process them to detect anomalies (e.g., fires). In such framework, recovery might be periodically performed in-network, while transmission to an external FC is necessary only when intervention is required.

DCS for WSNs without FC have other very interesting applications, e.g., distributed spectrum sensing in cognitive radio networks [14], [15] and wireless body area networks [16], [17], [18]. In the first scenario, one aims at estimating the occupied frequencies in cognitive radio networks, which can be done without FC, exploiting DCS and distributed recovery algorithms, as proposed in [14], [15] (here sparsity comes from the fact the spectrum is often under-used). In wireless body area networks, instead, very small sensors are applied on the body for health monitoring purposes, and connected wirelessly among them and to a FC (e.g., a smartphone). Applications of DCS have been studied in this case to increase efficiency and prolong network's lifetime [16], [17], while solutions without FC have been introduced very recently [18].

Distributed iterative algorithms for in-network recovery (without FC) of sparse signals have been lately proposed and theoretically analysed. In particular, there are methods based on iterative thresholding [8], [10], [11], [13], alternating direction method of multipliers [12], [15], and greedy algorithms [9]. In all these works, local cooperation is envisaged, that is, each node of the network can share information with its neighbors. Theoretical results and numerical simulations are convincing about the efficiency of these methods in terms of convergence to a stationary point and estimation accuracy.

However, a significant drawback of such in-network methods lies in the communication load: the local cooperation exploited for recovery generally requires many transmissions over the available communication links. In the mentioned literature, transmissions between neighboring occur at each iterative step, and the number of iterations may be very large, which entails significant energy consumption.

For this motivation, solutions to decrease the transmission load have been proposed, exploiting, e.g., randomization [10] and binary messaging [12], [13]. These schemes have been shown to dramatically reduce the number of required transmissions with respect to earlier methods. However, their efficiency has never been tested in real networks. Numerical tests in [10], [12], [13] demonstrate that the proposed communication protocols cut down the transmissions, but clearly do not

deal with real communications, synchronization, and storage problems. They assume, in fact, that (a) there is no loss of messages between nodes; (b) a global clock rules the updates of the nodes; (c) the nodes have sufficient memory to store the necessary data and perform computations.

The goal of this paper is to fill this gap, by proposing the first design and implementation of in-network sparse recovery methods in a real WSN. It is worth noting that practical implementations of DCS on WSNs have already been proposed in the literature, but only for the acquisition/compression phase (recovery is performed in a centralized manner by a powerful FC) [19], [20], [21], [22]. The novelty of our contribution consists in the design and implementation of both acquisition/compression and recovery phases on a real WSN, trying to cope with real transmissions and recovery complexity problems.

Specifically, we consider the algorithms proposed in [10] to solve a CS problem, and we demonstrate that they can work efficiently even on very low-power hardware. We build a testbed on STM32W boards [23], employing low-power and low-memory sensor nodes locally interconnected via wireless links, and we tailor the algorithms to handle the practical problems that are not envisaged in the theoretical/numerical analysis [10]. The algorithms in [10] have been conceived to reduce the number of transmissions, which is a first step towards practical feasibility. However, they rely on several assumptions, i.e., transmissions never fail and global synchronization rules the WSN, such that communications never collide. Moreover, the theoretical algorithms do not consider memory and computation restrictions for nodes. In practice, instead, WSN's nodes are endowed with very limited resources, e.g., a few kB of RAM and no floating point unit. In this paper, we address such complexity and storage problems, designing an efficient CS sensing matrix, which minimizes the occupied storage and simplifies the computations. Moreover, we manage the communication losses and the lack of global synchronization, by proposing an asynchronous scheme which is able to reduce the number of transmission collisions to reasonably small values.

The paper is organized as follows. In Section II, we review the implemented algorithms and the related theoretical aspects. In Section III, we describe our hardware and the WSN settings. Section IV presents our optimization tools to make the recovery efficient, while Section V illustrates the results of our real experiments, both on synthetic and physical signals. Finally, we draw some conclusions.

II. IMPLEMENTED RECOVERY ALGORITHMS

Let us consider a set $\mathcal{V} = \{1, 2, \ldots, V\}$ of nodes, interconnected via local communication links. Let us call $\mathcal{N}_v$ the neighborhood of node $v \in \mathcal{V}$, that is, the subset of nodes which can communicate with v. Each communication link is assumed valid in both directions, that is, the associated graph is undirected. Moreover, we assume that the graph is connected.

For our testbed, the goal of the network is to recover a k-sparse common signal $x \in \mathbb{R}^n$, that all the nodes acquire and compress individually as prescribed by CS theory, that is:

$$y_v = A_v x \in \mathbb{R}^m \quad m \ll n, \ v \in \mathcal{V} \quad (1)$$

where the sensing matrices $A_v \in \mathbb{R}^{m,n}$ can be different for each node (their structure will be discussed later). The goal of each $v \in \mathcal{V}$ is to recover x from y_v, leveraging only local communications. This model with common signal x could be easily extended to more complicated joint sparsity models and relative algorithms (e.g., JSM-2 [13] and JSM-3 [12]). However, even in JSM-2 and JSM-3, the estimation of the common component is the task that takes advantage of internode correlations, therefore our model captures the key point of the problem.

To this goal, we implement the asynchronous, broadcast, and gossip hard thresholding algorithms (AHT, BHT, and GHT, respectively) proposed in [11]. The common idea behind these algorithms is the following: each node of the network performs an iterative hard thresholding (IHT) [24], [25] introducing, at each step, information received from its neighbors. More precisely, let σ_k be the best k-term approximation of x:

$$\sigma_k(x) := \arg \min_{z \in \Sigma_k} \|x - z\|_2$$

where $\Sigma_k \subset \mathbb{R}^n$ is the subset of the k-sparse signals, and let $x_v(t)$ be the estimate of x held by v at time t. At each iteration step t, according to the chosen procedure, the following task is performed:

- AHT: a node v chosen uniformly at random wakes up, collects the estimates $x_w(t)$ of x from its neighbors $w \in \mathcal{N}_v$, and performs the update: $x_v(t + 1) = \sigma_k \left[\frac{1}{|\mathcal{N}_v|} \sum_{w \in \mathcal{N}_v} x_w(t) - \tau A_v^T (y_v - A_v x_v(t)) \right]$;
- BHT: a node v chosen uniformly at random wakes up, broadcasts its estimate $x_v(t)$ to its neighbors $w \in \mathcal{N}_v$, which perform the update: $x_w(t + 1) = \sigma_k \left[\frac{x_v(t) + x_w(t)}{2} - \tau A_w^T (y_w - A_w x_w(t)) \right]$;
- GHT: a pair of neighboring nodes (v, w) are chosen uniformly at random; v receives $x_w(t)$ and performs the update: $x_v(t + 1) = \sigma_k \left[\frac{x_v(t) + x_w(t)}{2} - \tau A_v^T (y_v - A_v x_v(t)) \right]$.

$\tau > 0$ is the gradient parameter; the procedure is repeated until convergence is obtained. AHT has been theoretically proved to converge; for BHT and GHT the fixed points have been theoretically characterized, while convergence has been shown via numerical simulations. We refer the interested reader to [11] for more details on convergence properties, estimation accuracy, and problems that the three algorithms can tackle (these are not limited to CS, but envisage wider sparsity constrained optimization models).

III. WSN TESTBED

In this section, we illustrate the architecture of the WSN that we use in our experiments.

A. Hardware

We use two types of boards, MB851 and MB954 from ST Microelectronics [23]. Especially designed for WSNs, these boards are composed by a series of peripherals, such as GPIO ports, LEDs, a temperature sensor, and a STM32W SoC, which provides both networking capabilities and computational engine. This unit consists of a microcontroller with a 32 bit, 24 MHz, ARM Cortex-M3 CPU, a 265 kB eFlash, 8 kB RAM memory, and a IEEE 802.15.4, 2.4 GHz radio transmitter.

B. Contiki OS

The sensors are equipped with Contiki, a lightweight event driven open source operating system especially designed for the Internet of Things, whose main purpose is to manage connectivity for low cost small devices, such as our sensors. Contiki provides a platform independent high level programming framework, based on C language, coming with lots of APIs to easily control the peripherals, as well as the network stack. Moreover, it provides several options for adding software features, e.g., a multi-threaded environment, a graphical user interface, a file system. Our AHT, BHT and GHT applications are generally organized as follows. On one hand, a client process is used to perform the update phase, during which the node can send or receive the estimated signals from its neighbors, according to the specific algorithm implementation. On the other hand, a server process is used to wait for neighbors TCP connections. During this phase, nodes can be actually contacted by neighbors. This mechanism is independent from the specific algorithm. In AHT and GHT, nodes connect to their neighbors and ask for estimates. In BHT, nodes perform an active connection and spontaneously send their estimates. An event driven mechanism is used to exchange information between the client ad server process in order to update reconstructed signals before each iteration.

IV. REAL RECOVERY IN WSN

Using the hardware and software described in the previous section, we implement AHT, BHT, and GHT to perform on-field tests and evaluate their practical feasibility in terms of WSN computational capabilities, communication protocols, and energy consumption. We assume that each node $v \in \mathcal{V}$ acquires a sparse signal $x \in \mathbb{R}^n$, and then compresses it via its own sensing matrix $A_v \in \mathbb{R}^{m,n}$, obtaining the compressed measurements vector $y_v = A_v x$.

Two main issues arise in this real implementation. The first one is due to scarce memory: for compression and reconstruction, the sensing matrix $A_v \in \mathbb{R}^{m \times n}$, which may have considerable size, has to be stored in the RAM of the node. From CS theory, we know that random Gaussian, Rademacher, circulant matrices [26] are suitable for reconstruction, with slight different recovery performance. Different strategies can be used to store efficiently such matrices, according to the specific purpose. The extreme strategies are (a) store only the seed and regenerate the matrix each time it is needed; (b) generate the matrix once and store it. Clearly, (a) is optimal from the memory occupation perspective, but requires

a number of extra computations in the recovery phase; (b) minimizes the number of computations (and consequently used energy) but requires considerable memory. For our setting, in which both memory and computational capabilities are scarce, after some tests, we have found that Rademacher, circulant matrices are a good tradeoff. Rademacher matrices have entries equal to ± 1 (drawn uniformly at random); circulant matrices are obtained by repeated circular shifts of the first row. Therefore, a binary vector of length n then is sufficient to store a Rademacher, circulant matrix. Hence, in our experiments, each node generates its own Rademacher, circulant matrix once and then keeps it stored in its RAM.

The second issue is related to the algorithms. In [11], a common clock is assumed to be available in the network, so that nodes wake up exclusively and uniformly at random. This is a technical assumption that simplifies the theoretical analysis, but in the practice, global synchronization is difficult to maintain. Therefore, in our implementation nodes wake up and communicate at random time instants, each of them with its own clock. This may clearly cause collisions, loss of messages, and reconstruction latency, whose consequences have to be evaluated in our tests.

V. EXPERIMENTS

In this section, we present our in-field experiments in a WSN composed by 5 nodes as described in Section III. We illustrate the cases of complete and ring topologies, which respectively represent the most and the least connected ones among the connected topologies. For each node $v \in \mathcal{V} = \{1, \ldots, 5\}$ and its final estimate $\hat{x}_v$, we define the estimation error as

$$e_v = \frac{\|\hat{x}_v - x\|_2^2}{\|x\|_2^2}.$$

A. Synthetic signals

In the first experiment, we generate a synthetic signal of length $n = 100$ and sparsity $k = 5$. Non-zero entries are generated uniformly at random in $[-3, 3]$. The signal is sent to all the nodes, which compress it to as in (1) to length $m = 20$. At the end of the reconstruction, we recollect the reconstructed signals for offline analysis. 5 runs are performed for each algorithm.

In Figure 1, we show the estimation error of AHT, BHT, GHT for each run and each node, assuming a complete topology. In Figure 2, we show the corresponding average communication error percentages, that is, the rate of lost transmissions. We observe that the reconstruction error is small (at most of order 10^{-4}). The average communication error percentage is 2% for AHT, while it is null for GHT. This makes sense because GHT involves only one link at any update, which reduces the probability of collisions.

In Figure 3, the error is shown in case of ring topology. The error is about 10^{-4} for AHT, and 10^{-5} for BHT and GHT. The average communication error percentages (Figure 4) are below 0.5% for AHT and BHT, and null for GHT (again this is due to the fact that GHT uses only one communication link
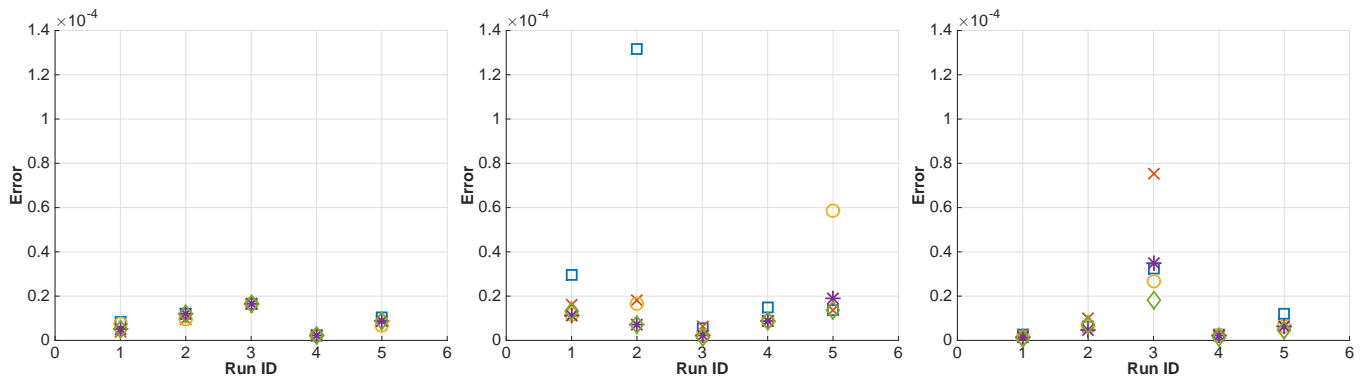
Figure 1. Synthetic signals, complete topology: error for AHT, BHT, and GHT (from left to right)
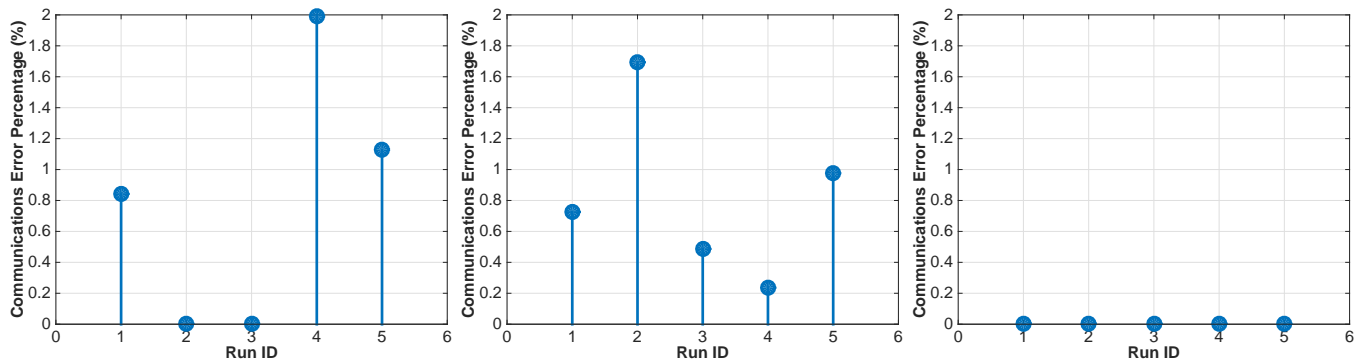

Figure 2. Synthetic signals, complete topology: average communication error percentages for AHT, BHT, and GHT (from left to right)
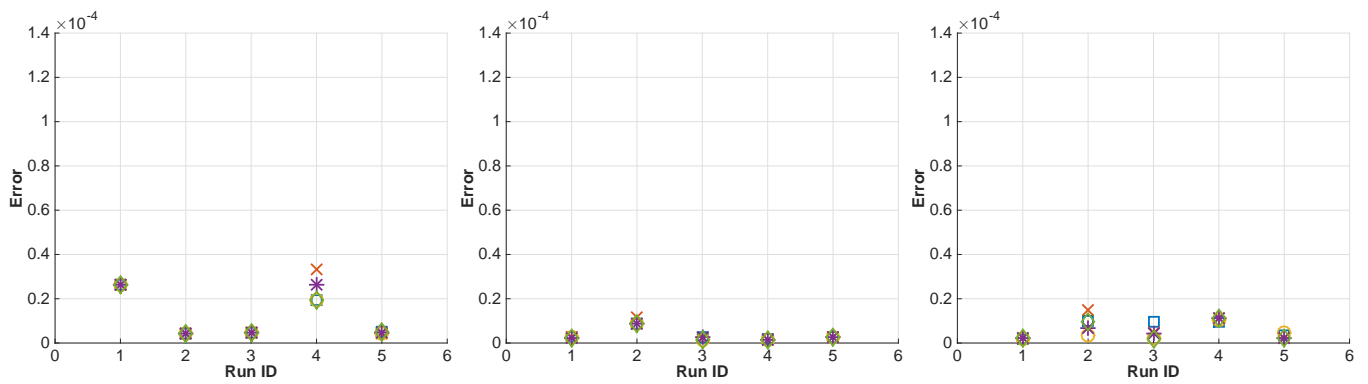

Figure 3. Synthetic signals, ring topology: errors for AHT, BHT, and GHT (from left to right), in 5 different runs. Different markers indicate different nodes.
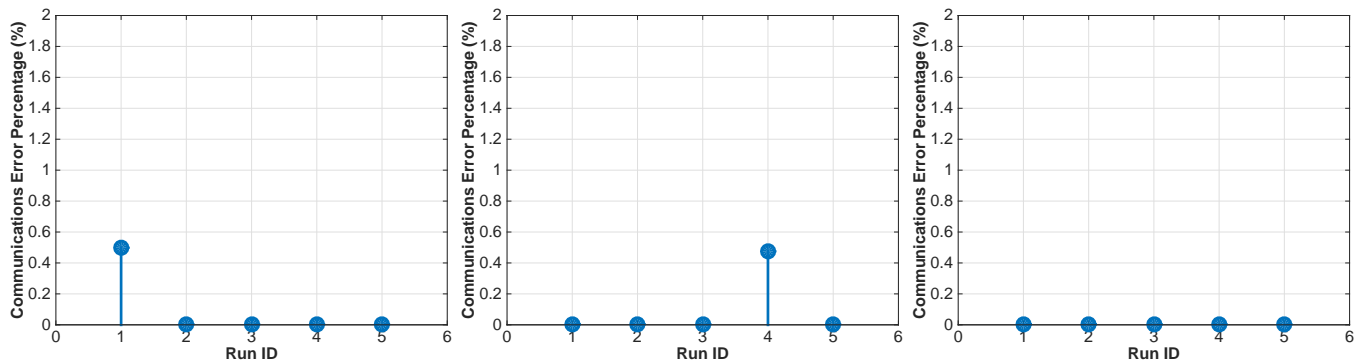

Figure 4. Synthetic signals, ring topology: average communication error percentages for AHT, BHT, and GHT (from left to right)

at each iteration). We observe that the communication error percentages for AHT and BHT are smaller for ring topologies with respect to complete topologies, which is expected because in complete topologies all the links are used at each update, as the neighborhood of each node is the whole network.

To obtain these estimates, in both topologies, the algorithms require about 20, 40, and 20 iterations for AHT, BHT and GHT respectively. The average reconstruction time is 4 minutes for AHT and BHT, and reduces to 1 minute for GHT. AHT and BHT differ in the number of iterations, but the average reconstruction time is quite similar. This is due to the doubled update rate of BHT, where nodes compute a gradient step on their own after having broadcast their estimated signal to their neighbors.

B. Real signals: temperature in a room

As the used nodes embed a temperature sensor, we conduct a test in which the WSN aims to recover the average temperature in a room. Each node is deployed in a different position of the room, and small differences are then expected in the sensed temperature values. We implement AHT, BHT, and GHT to estimate the mean temperature in a collaborative way. The sparsity here resides in the difference signal: we assume that each node takes $n = 50$ temperature measurements $x_1, \ldots, x_n$ at uniform unitary time intervals and iteratively computes the difference vector $(x_1, x_2 - x_1, \ldots, x_n - x_1)$. In absence of abrupt changes, the temperature is almost constant, hence the difference vector has non-exact sparsity $k = 1$. By non-exact sparsity we mean that most of entries can be approximated to zero, but are not exactly null. This is a typical real, noisy model. After acquisition, the node compresses its temperature readings via its Rademacher, circulant matrix to $m = 5$ measurements. Once finished, the sensors send both the initial temperature signal measured and its reconstructed version for comparison purposes.

In Figure 5 and 6, we show the results of such an experiment, with complete and ring topologies, respectively. Dealing with noise, we let the sensors perform 20 more iterations with respect to the corresponding synthetic signal scenario. We also depict the absolute value between the average temperature sensed by all the nodes and their respective reconstructed signals. Here we can see very small differences, as the error is almost always smaller than 0.5 °C.

C. Energy consumption considerations

In this section, we analyze the energy consumption based on current and energy measurements.

In the first test, we equip the sensors with new sets of batteries and we monitor the time needed to deplete them. In this configuration the boards are programmed to keep on reconstructing a synthetic signal of length 100 for an infinite number of AHT iterations. We observe approximately 39 hours of continuous work for MB851 boards, and 32 hours for MB954 boards. This difference in duration time is caused by the power amplifier mounted by the latter type of boards, which requires higher energy amounts. This aspect should

also highlight the large weight of radio transmissions in the overall energy consumption, which should definitely clarify the need for drastically reducing the amount of communications among nodes in order to make DCS appealing in real WSN contexts. The selected sensor parameter settings cause them to perform 6 iterations per minute on average. This means MB954 can iterate our algorithms approximately 11520 times before needing to be fed with new batteries, while MB851 is able to last for about 14040 iterations. These numbers are definitely larger than the observed number of iterations per reconstruction, hence the boards can performs many runs before needing batteries substitution.

We obtain a more in depth analysis of energy consumption by measuring the energy absorbed by the boards. For this purpose, we measure the current passing through the jumper closing the circuit in battery mode. The current has been measured both through a digital multimeter and an oscilloscope for better analyzing the impact of data transmissions on the overall power absorption. Tests have demonstrated an average 25 mA current during non-transmission interval, and an increment in current flow during data exchanges. In particular, during transmissions the total current increases to 30 mA for MB851 boards, and 160 mA for MB954 ones. Each transmission lasts up to 8 ms and 2 ms on MB851 and MB954 boards, respectively. Using 3V batteries, the total energy consumption during a transmission is about 7×10^{-4} J and 9×10^{-4} J for MB851 and MB954 boards, respectively. The differences are again due to the presence of a power amplifier on MB954 boards.

We finally notice that no remarkable current variations are observed through the oscilloscope when nodes perform computations. This aspect confirms the extremely low impact of calculations in our setup, obtained through the use of Rademacher sensing matrices: hardware multiplications by -1 and +1 can be easily implemented, letting us save energy, as also noticed in [19], [20].

VI. CONCLUDING REMARKS

In this paper, we have proposed the first real implementation of distributed sparse recovery algorithms in WSNs. We have built a basic WSN, with very low-power and low-memory hardware and we have implemented the algorithms proposed in [11], based on iterative hard thresholding, to solve a DCS problem. The result is encouraging: even though computational capabilities are small and communications are not always successful, we are able to accurately estimate the signals and consensus is achieved between nodes. Tests have been conducted both on synthetic and real temperature signals measured by the WSN itself. Next tests will be conducted on larger WSNs, and using different algorithms and sparsity models, e.g., [13].
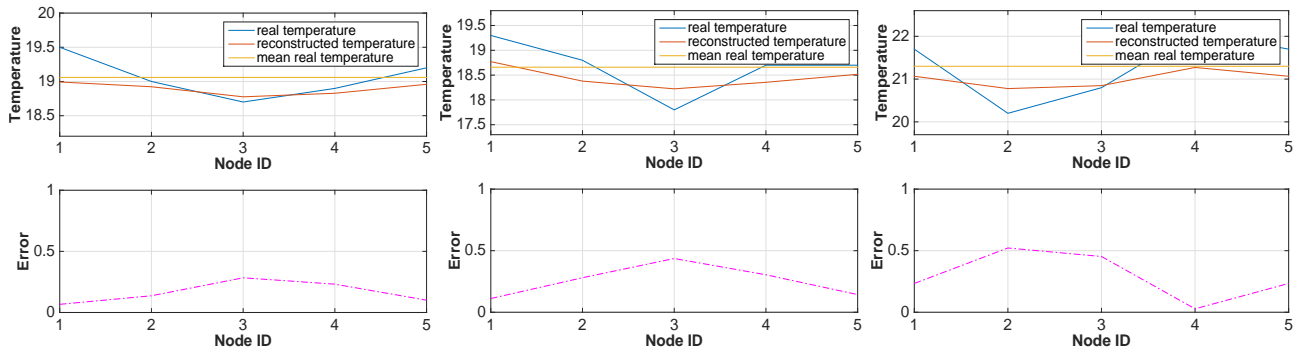
Figure 5. Average temperature estimate (°C) for AHT, BHT, and GHT (from left to right); complete topology
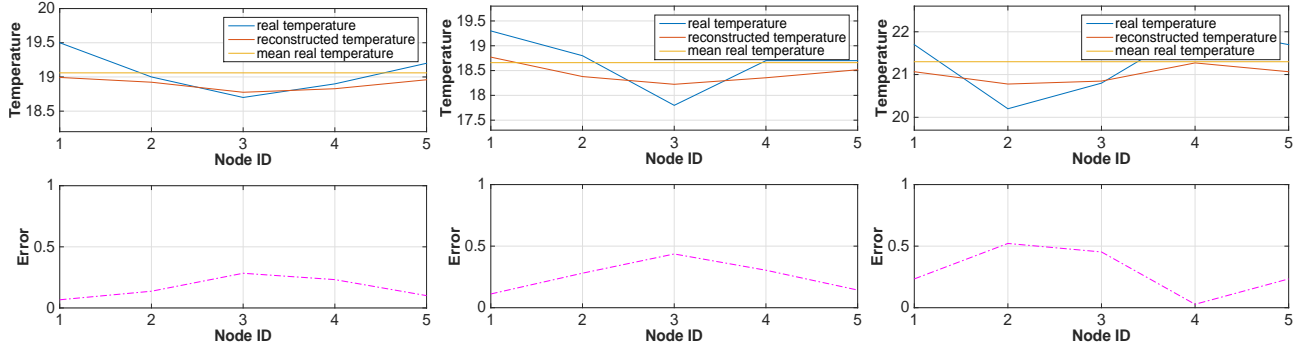

Figure 6. Average temperature estimate (°C) for AHT, BHT, and GHT (from left to right); ring topology

REFERENCES

[1] G. Coluccia, C. Ravazzi, and E. Magli, *Compressed Sensing for Distributed Systems*. Springer Singapur, 2015.

[2] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, pp. 1289 – 1306, 2006.

[3] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Proc. Magazine*, vol. 25, pp. 21 – 30, 2008.

[4] M. A. Davenport, M. F. Duarte, Y. C. Eldar, and G. Kutyniok, *Introduction to Compressed Sensing*. Cambridge University Press, 2012.

[5] D. Baron, M. F. Duarte, M. B. Wakin, S. Sarvotham, and R. G. Baraniuk, "Distributed compressive sensing of jointly sparse signals," in *Asilomar Conf. Signals, Sys., Comput.*, 2005, pp. 1537–1541.

[6] D. Baron, M. F. Duarte, S. Sarvotham, M. B. Wakin, and R. G. Baraniuk, "An information theoretic approach to distributed compressed sensing," in *Allerton Conf. Comm., Cont., Comp.*, 2005.

[7] M. F. Duarte, S. Sarvotham, D. Baron, M. B. Wakin, and R. G. Baraniuk, "Distributed compressed sensing of jointly sparse signals," in *Proc. 39th Asilomar Conf. Signals, Systems and Computers*, 2005.

[8] S. Patterson, Y. Eldar, and I. Keidar, "Distributed compressed sensing for static and time-varying networks," *IEEE Trans. Signal Process.*, vol. 62, no. 19, pp. 4931 – 4946, 2014.

[9] T. Wimalajeewa and P. Varshney, "OMP based joint sparsity pattern recovery under communication constraints," *IEEE Trans. Signal Process.*, vol. 62, no. 19, pp. 5059–5072, Oct 2014.

[10] C. Ravazzi, S. M. Fosson, and E. Magli, "Distributed iterative thresholding for l_0/l_1-regularized linear inverse problems," *IEEE Trans. Inf. Theory*, vol. 61, no. 4, pp. 2081 – 2100, 2015.

[11] ——, "Randomized algorithms for distributed nonlinear optimization under sparsity constraints," *IEEE Trans. Signal Process.*, vol. 64, no. 6, pp. 1420 – 1434, 2016.

[12] J. Matamoros, S. M. Fosson, E. Magli, and C. Antón-Haro, "Distributed ADMM for in-network reconstruction of sparse signals with innovations," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 1, no. 4, pp. 225 – 234, 2015.

[13] S. M. Fosson, J. Matamoros, C. Anton-Haro, and E. Magli, "Distributed recovery of jointly sparse signals under communication constraints," *IEEE Trans. Signal Process.*, vol. 64, no. 13, pp. 3470–3482, 2016.

[14] J. A. Bazerque and G. B. Giannakis, "Distributed spectrum sensing for cognitive radio networks by exploiting sparsity," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1847–1862, 2010.

[15] F. Zeng, C. Li, and Z. Tian, "Distributed compressive spectrum sensing in cooperative multihop cognitive networks," *IEEE J. Sel. Top. Sign. Proces.*, vol. 5, no. 1, pp. 37–48, 2011.

[16] H. Mamaghanian, N. Khaled, D. Atienza, and P. Vandergheynst, "Compressed sensing for real-time energy-efficient ECG compression on wireless body sensor nodes," *IEEE Trans. Biomed. Eng.*, vol. 58, no. 9, pp. 2456–2466, 2011.

[17] A. Ravelomanantsoa, H. Rabah, and A. Rouane, "Simple and efficient compressed sensing encoder for wireless body area network," *IEEE Trans. Instrum. Meas.*, vol. 63, no. 12, pp. 2973–2982, 2014.

[18] A. Bertrand, "Distributed signal processing for wireless eeg sensor networks," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 23, no. 6, pp. 923–935, 2015.

[19] D. Brunelli and C. Caione, "Sparse recovery optimization in wireless sensor networks with a sub-nyquist sampling rate," *Sensors*, vol. 15, no. 7, pp. 16 654–16 673, 2015.

[20] C. Caione, D. Brunelli, and L. Benini, "Compressive sensing optimization for signal ensembles in wsns," *IEEE Trans. Ind. Informat.*, vol. 10, no. 1, pp. 382–392, 2014.

[21] ——, "Distributed compressive sampling for lifetime optimization in dense wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. 8, no. 1, pp. 30–40, Feb 2012.

[22] W. Chen and I. Wassell, "Energy-efficient signal acquisition in wireless sensor networks: a compressive sensing framework," *IET Wireless Sensor Systems*, vol. 2, no. 1, pp. 1–8, 2012.

[23] STMicroelectronics, "STM32W datasheet http://www.st.com/stm32w."

[24] T. Blumensath and M. E. Davies, "Iterative thresholding for sparse approximations," *J. Fourier Anal. Appl.*, vol. 14, no. 5-6, pp. 629 – 654, 2008.

[25] ——, "Iterative hard thresholding for compressed sensing," *Appl. Comput. Harmon. Anal.*, vol. 27, no. 3, pp. 265–274, 2009.

[26] H. Rauhut, "Compressive sensing and structured random matrices," *Theoretical foundations and numerical methods for sparse recovery*, vol. 9, pp. 1–92, 2010.