Distributed Non-Asymptotic Confidence Region Computation over Sensor Networks

Vincenzo Zambianchi, Francesca Bassi, Alex Calisti, Davide Dardari, Michel Kieffer, Gianni Pasolini

Distributed Non-Asymptotic Confidence Region Computation over Sensor Networks

V. Zambianchi, *Student Member, IEEE,* F. Bassi, *Member, IEEE,*
A. Calisti, *Student Member, IEEE,* D. Dardari, *Senior Member, IEEE,*
M. Kieffer, *Senior Member, IEEE,* and G. Pasolini, *Member, IEEE,*

Abstract

This paper addresses the distributed computation of exact, non-asymptotic confidence regions for the parameter estimation of a linear model from observations at different nodes of a network of sensors. If a central unit gathers all the data, the sign perturbed sums (SPS) method proposed by Csáji *et al.* can be used to define guaranteed confidence regions with prescribed confidence levels from a finite number of measurements. SPS requires only mild assumptions on the measurement noise. This work proposes distributed solutions, based on SPS and suited to a wide variety of sensor networks, for distributed in-node evaluation of non-asymptotic confidence regions as defined by SPS. More specifically, a Tagged and Aggregated Sum information diffusion algorithm is introduced, which exploits the specificities of SPS to avoid flooding the network with all measurements provided by the sensors. The performance of the proposed solutions is evaluated in terms of required traffic load, both analytically and experimentally on different network topologies. The best information diffusion strategy among nodes depends on how structured the network is.

Index Terms

Confidence region, guaranteed precision, distributed algorithms, sign-perturbed sums

V. Zambianchi, A. Calisti, D. Dardari, and G. Pasolini are with the Department of Electrical, Electronics and Information Engineering “G. Marconi” (DEI), University of Bologna, via Venezia 52, Cesena (FC) 47521, Italy. E-mail: {vincenzo.zambianchi, alex.calisti, davide.dardari, gianni.pasolini}@unibo.it. M. Kieffer is with Laboratoire de Signaux et Systèmes (L2S) CNRS-CentraleSupelec-Université Paris Sud, Gif-sur-Yvette, France. E-mail: michel.kieffer@lss.supelec.fr. F. Bassi is with ESME Sudria, Ivry-sur-Seine, France and with Laboratoire de Signaux et Systèmes (L2S), Gif-sur-Yvette, France. E-mail: francesca.bassi@lss.supelec.fr. This work has been supported by the EU funded Newcom# Project, contract n.318306, and in part by the European H2020 project XCycle (Grant 635975). A fraction of the presented results are published in [1].

I. Introduction

A Sensor Network (SN) consists of energy-limited sensing devices deployed to collaborate in performing a common task. Examples may be the monitoring of an environmental parameter (*e.g.*, temperature or pressure [2]–[4]), the detection of a binary event [5], the estimation of a spatial field [6], the estimation of the coordinates of a signal source [7], etc. Depending on the specific task requirements (fault tolerance, privacy issues, energy constraints), either a centralized or a distributed approach can be adopted. In the centralized setup a central unit collects all the information and completes the task, whereas in the distributed setup the nodes exchange information and accomplish the task locally.

As far as the centralized estimation of physical parameters is concerned, maximum likelihood (ML) or least squares (LS) estimators [8] can be adopted, both working under the hypothesis of having all the required observations available at one central unit.

However, the scarce robustness to central unit failures and poor network scalability have brought to consideration distributed approaches. For instance, recursive weighted LS estimation has been considered [9], [10], alongside a consensus-based algorithm that allows to incorporate information from neighbor nodes in the local estimate. A similar approach is taken within the Bayesian framework [11]–[13], where consensus-based distributed Kalman filtering is proposed.

Whatever the adopted processing strategy, either centralized or distributed, in many applications a simple point estimate of the parameter vector of interest is not sufficient if not associated with a confidence region to assess the estimation uncertainty. Classically, the estimation accuracy is investigated using Cramér-Rao-like bounds [14]–[17]. Confidence regions can also be derived as a by-product of distributed Kalman filtering [12], [13]. Nevertheless, strong assumptions on the measurement noise (typically Gaussian) are necessary and most of the techniques provide only approximate, possibly asymptotically tight, confidence regions.

In centralized setups, provided that the regression model is linear, the derivation of confidence regions in the non-asymptotic regime is possible using the results in [18]–[24]. The Leave-out Sign-dominant Correlation Regions (LSCR) method [18], [19] and the sign perturbed sums (SPS) method [20], [22] allow the central unit to derive, from a finite set of measurements, *guaranteed*, *non-asymptotic* confidence regions with prescribed confidence levels around the LS estimate of the parameter vector. Differently from Cramér-Rao-like bounds, SPS does not require precise statistical knowledge of the noise, and works under mild assumptions on its distribution [22], [23]. Efficient centralized characterization of confidence

regions can be obtained using interval analysis [24].

A. Main Contributions

In [1] we showed that confidence regions, as defined by SPS, may be evaluated in a distributed way, for example in wireless sensor networks (WSNs). For that purpose, the nodes share their local information with each other and the confidence region computation is performed locally. Three information diffusion approaches (data flooding and parallel in-node processing, distributed processing via average consensus, and mixed flooding+consensus) have been considered in [1] to provide each node with the information allowing a distributed computation of the confidence region.

In all cases, the information diffusion strategy, in addition to the network topology, determines the amount of data exchanged, which needs to be restrained.

In this regard, a novel information diffusion strategy, named Tagged and Aggregated Sums (TAS), is presented in this paper. It exploits the peculiarities of the SPS method, leading to a reduction of the amount of information to be exchanged among nodes and, at the same time, it is sufficiently general to be applied to any network topology. It is compared with classical *general purpose* information diffusion strategies, such as flooding [2], [25] and consensus algorithms [11], in terms of generated traffic load as well as of confidence region volume/traffic trade-off. Performance predictions, simulation and experimental results are provided for various topologies, extending preliminary results presented in [1].

B. Organization of the paper

The remainder of this work is organized as follows. Section II formulates the confidence region computation problem and recalls the SPS method. Section III presents several information diffusion strategies. Information diffusion techniques are compared on various network topologies in Sections IV and V. Experimental results presented in Section VI allow to account for MAC layer aspects and confirm that the best information diffusion strategy depends on the way the SN is structured. Conclusions are drawn in Section VII.

II. Non-asymptotic confidence regions

In this paper vectors are denoted by bold lowercase letters while matrices are indicated with bold capital letters. For the reader's convenience, the most significant symbols introduced in the following and their meaning are reported in Table I.

TABLE I

TABLE OF SYMBOLS AND RELATED MEANINGS

Linear regression system	
$\boldsymbol{n}_{\mathrm{p}}$	dimension of the parameter vector
Θ	parameter space ($\Theta \subset \mathbb{R}^{n_{\mathrm{p}}}$)
$\boldsymbol{\theta}$	vector belonging to the parameter space Θ
$\boldsymbol{\theta}^*$	true value of the $n_{\mathrm{p}} \times 1$ parameter vector
$\widehat{\boldsymbol{\theta}}$	least squares estimate of $\boldsymbol{\theta}^*$
$\mathbf{x}_i$	location of Node i
$\boldsymbol{\varphi}_i$	regressor vector at $\mathbf{x}_i$;
y_i	measurement collected by Node i
SPS variables	
m	amount of sums considered by the SPS method
$a_{j,k}$	realizations of independent random signs
$\mathbf{Q}_n$	SPS normalization matrix
$\mathbf{s}_0(\boldsymbol{\theta})$	unperturbed sum
$\mathbf{s}_j(\boldsymbol{\theta})$	$m-1$ sign perturbed sums ($j = 1, ..., m-1$)
$\boldsymbol{\Sigma}_q$	non-asymptotic confidence region
TAS information diffusion algorithm	
$\mathbf{t}_r^{(k)}$	tag vector to be transmitted by Node k in round r
$\mathbf{d}_r^{(k)}$	dataset to be transmitted by Node k in round r
d_{TAS}	size of the dataset transmitted by TAS
$\boldsymbol{\delta}_{i,j,..}$	dataset with sums involving data from Nodes i, j, ...
$\boldsymbol{\delta}_{\mathrm{F}}^k$	dataset at Node k after final wrap-up
$n_{\mathrm{TAS}}^{\mathrm{GT}}$	amount of data transmitted by TAS in a generic tree
$n_{\mathrm{TAS}}^{\mathrm{BT}}$	amount of data transmitted by TAS in a binary tree
$n_{\mathrm{TAS}}^{\mathrm{CN}}$	amount of data transmitted by TAS in a clustered network
Flooding information diffusion algorithm	
d_{F}	size of the dataset transmitted by flooding (FL)
$n_{\mathrm{FL}}^{\mathrm{GT}}$	amount of data transmitted by FL in a generic tree
$n_{\mathrm{FL}}^{\mathrm{BT}}$	amount of data transmitted by FL in a binary tree
$n_{\mathrm{FL}}^{\mathrm{CN}}$	amount of data transmitted by FL in a clustered network
Network setup	
n	number of nodes in the network
$\mathcal{N}(k)$	set of neighbors of node k
$\lambda(\ell)$	number of nodes at Level ℓ (tree network)
$\overline{\lambda}(\ell)$	number of nodes with no children at level ℓ (tree netw.)
L	number of levels of the tree network (excluding the root)
n_{c}	number of clusters in the clustered network
n_i^{c}	number of nodes (clusterhead included) in the i-th cluster

A. Problem Formulation

Consider some spatial field described by the following parametric model [26]

$$y\left(\mathbf{x},\boldsymbol{\theta}\right)=\boldsymbol{\varphi}^{T}\left(\mathbf{x}\right)\boldsymbol{\theta}, \tag{1}$$

where $\mathbf{x}\in\mathbb{R}^{n_x}$ is some vector of experimental conditions (time, location…) under which the field is observed, $\boldsymbol{\varphi}\left(\mathbf{x}\right)$ is the regressor function, and $\boldsymbol{\theta}$ is the vector of unknown parameters. Measurements are taken by a network of n sensor nodes, spread at random locations $\mathbf{x}_i\in\mathbb{R}^{n_x}$, $i=1,\ldots,n$. Node i collects the scalar measurement y_i according to the local measurement model

$$y_i=y\left(\mathbf{x}_i,\boldsymbol{\theta}^*\right)+w_i=\boldsymbol{\varphi}_i^{T}\boldsymbol{\theta}^*+w_i, \tag{2}$$

where $\boldsymbol{\varphi}_i=\boldsymbol{\varphi}\left(\mathbf{x}_i\right)$ is the $n_{\mathrm{p}}\times 1$ regressor vector at $\mathbf{x}_i$; $\boldsymbol{\theta}^*$ is the true value of the deterministic $n_{\mathrm{p}}\times 1$ parameter vector, which is only known to belong to the subset $\Theta\subset\mathbb{R}^{n_{\mathrm{p}}}$; w_i represents the measurement noise at Node i.

As in [22], the random variables with realizations w_i, $i=1\ldots,n$ are assumed to be statistically independent and to follow a symmetrical distribution.[1] Deterministic regressors $\boldsymbol{\varphi}_i$ are considered here, but this work may be extended to the case of random exogenous regressors, *i.e.*, regressors $\boldsymbol{\varphi}_i$s that are independent on the noise terms. We consider the worst case in which the value of $\boldsymbol{\varphi}_i$ is assumed known only by Node i. Moreover, we assume that there exists $n'<n$ such that for all subset of indexes $\mathcal{I}\subset\{1,\ldots,n\}$ with $|\mathcal{I}|\geqslant n'$, the regressors are such that $\det\mathbf{Q}_{\mathcal{I}}\neq 0$, where

$$\mathbf{Q}_{\mathcal{I}}=\frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}}\boldsymbol{\varphi}_i\boldsymbol{\varphi}_i^{T}. \tag{3}$$

In what follows, $\mathbf{Q}_{\{1,\ldots,n\}}$ is denoted $\mathbf{Q}_n$. The purpose of the network is to let each node capable of computing locally the confidence region of the estimate of $\boldsymbol{\theta}^*$ with the lowest impact on network traffic.[2]

B. Centralized SPS

The centralized SPS method [20], [22] assumes all measurements and regressors to be known at the central processing unit. It defines an exact confidence region around the least

[1]In [23], no symmetry condition is considered, the random measurement sequence is only assumed to form an exchangeable sequence of random variables. This work readily extends to this alternative assumption.

[2]The proposed approach readily extends to vector fields in which the measurement is a vector, as well as to vectors of measurements, provided that the noise components of each vector are independent and symmetrically distributed. This extension is not considered here to lighten notations.

squares estimate $\widehat{\boldsymbol{\theta}}$ of $\boldsymbol{\theta}^*$, obtained as the solution of the normal equations $\sum_{k=1}^{n} \boldsymbol{\varphi}_k \left(y_k - \boldsymbol{\varphi}_k^T \boldsymbol{\theta}\right) = \mathbf{0}$. For that purpose, as in [22], consider the *unperturbed sum* as the following function over Θ

$$\mathbf{s}_0(\boldsymbol{\theta}) = \mathbf{Q}_n^{-1/2} \sum_{k=1}^{n} \boldsymbol{\varphi}_k \left(y_k - \boldsymbol{\varphi}_k^T \boldsymbol{\theta}\right) \tag{4}$$

and the $m-1$ *sign-perturbed sums*, defined $\forall j = 1, \ldots, m-1$ as the following functions over Θ

$$\mathbf{s}_j(\boldsymbol{\theta}) = \mathbf{Q}_n^{-1/2} \sum_{k=1}^{n} a_{j,k} \boldsymbol{\varphi}_k \left(y_k - \boldsymbol{\varphi}_k^T \boldsymbol{\theta}\right), \tag{5}$$

where $a_{j,k} \in \{\pm 1\}$ are realizations of independent random signs.[3] For each $\boldsymbol{\theta} \in \Theta$, one considers the elements of the set

$$\mathcal{Z}(\boldsymbol{\theta}) = \left\{ z_j(\boldsymbol{\theta}) = ||\mathbf{s}_j(\boldsymbol{\theta})||_2^2 \right\}_{j=0,1,\ldots,m-1}, \tag{6}$$

and lists them in increasing order, giving rise to a permutation $\pi_{\boldsymbol{\theta}}(\cdot) : \{0, \ldots, m-1\} \rightarrow \{0, \ldots, m-1\}$. One defines the set

$$\boldsymbol{\Sigma}_q = \left\{ \boldsymbol{\theta} \in \Theta \mid \pi_{\boldsymbol{\theta}}(0) \leq m - 1 - q \right\} \tag{7}$$

which contains all $\boldsymbol{\theta} \in \Theta$ for which the rank of $z_0(\boldsymbol{\theta})$ in the ordering is among the $m-q$ smallest, with $q = 1, ..., m-1$. In [20], [22], it was proven that

$$\text{Prob}(\boldsymbol{\theta}^* \in \boldsymbol{\Sigma}_q) = 1 - \frac{q}{m}. \tag{8}$$

As a consequence $\boldsymbol{\Sigma}_q$ is a non-asymptotic confidence region with *exact* confidence level $1 - q/m$. The values of q and m may be chosen to get the requested confidence level of the confidence region $\boldsymbol{\Sigma}_q$ for the estimate $\widehat{\boldsymbol{\theta}}$ of $\boldsymbol{\theta}^*$.

An extension of the SPS method is presented in [23], which considers that $\pi_{\boldsymbol{\theta}}$ is one of the $m!$ possible permutations on $\mathcal{Z}(\boldsymbol{\theta})$. Letting Π_k be a set of k permutations, the set

$$\boldsymbol{\Sigma}_k = \left\{ \boldsymbol{\theta} \in \Theta \mid \pi_{\boldsymbol{\theta}} \in \Pi_k \right\} \tag{9}$$

is defined, which allows one getting confidence regions such that

$$\text{Prob}(\boldsymbol{\theta}^* \in \boldsymbol{\Sigma}_k) = \frac{k}{m!}. \tag{10}$$

Notice that (8) and (10) are equivalent for $k = m! - q(m-1)!$.

The main advantage of the extension of SPS in [23] over that in [22] is that in the former the resolution of the confidence level is $1/m!$, while in the latter it is $1/m$. For example, with the

[3]A random sign is a symmetric ± 1 value random variable taking both values with the same probability.

approach in [23], confidence regions for levels $\{100\%, 96\%, \ldots, 62.5\%, \ldots, 8.3\%, 4.2\%\}$ may be theoretically defined for $m = 4$, whereas confidence regions only for levels $\{100\%, 75\%, 50\%, 25\%\}$ are defined in [22]. This difference may appear to be interesting when SPS is used in a distributed version, where small values of m are of interest, to restrain communication costs. Nevertheless, our experiments show that with the approach in [23], when choosing $k \geqslant m! - (m-1)!$, the confidence regions are not necessarily compact.

Non-asymptotic confidence regions as defined in [22] may be outer-approximated using ellipsoids, as in [22], boxes, or union of non-overlapping boxes as in [24].

In the following, the distributed computation of $\boldsymbol{\Sigma}_k$ is addressed considering different information diffusion strategies.

III. Information Diffusion Algorithms

This section describes the distributed computation of confidence regions as defined by the SPS algorithm [22]. Concurrent procedures for information diffusion applicable to any network topology are considered. The purpose is that each node collects the largest amount of information with the lowest amount of data exchanged in the network so that it is able to compute locally the confidence region of the LS estimate for any $\boldsymbol{\theta}^*$.

Before entering into the details of our investigation, few words are needed to clarify the different roles played by the *physical*, *logical*, and *processing* elements that affect the performance of the investigated strategies.

The *physical* element of a SN is given by the deployment of nodes in the given scenario, that defines the network layout. On this regards, the only condition we assume is that all nodes can communicate with each other, either with single or multi-hop links.

Given the network layout, a routing protocol is typically applied, which defines the *logical topology* of the network, that is, the set of paths and directions which data can flow through. On top of the same network layout, in fact, different kinds of *logical topologies* can be created, either hierarchical (tree topology, cluster topology…) or flat, depending on the routing protocol that defines, in other words, the possible *information paths* for the given deployment of nodes.

Finally, the information diffusion strategies investigated in this paper concern the *processing* elements. In fact, they deal with the way the information is managed (aggregated and/or fused) by a node before being transmitted to the next one(s) according to the *logical topology*. A node can transmit, for instance, either elementary data (as done by FL) or a processed version of data (as done by consensus schemes and the proposed TAS algorithm).

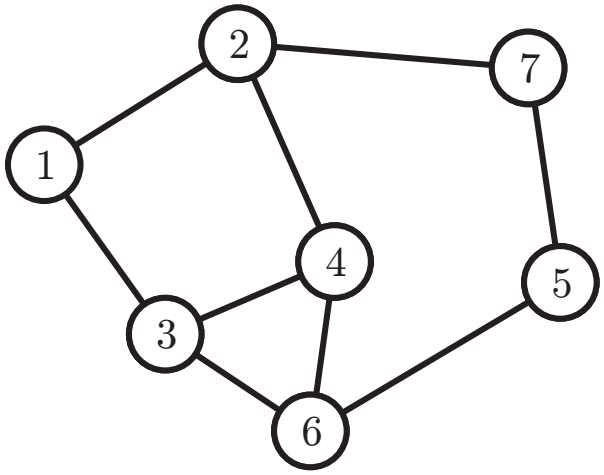

Fig. 1. Toy network example

Obviously, given a fixed logical network topology, it is always possible to design an ad-hoc information diffusion algorithm that provides the best performance. However, we are interested in designing procedures that are not tailored to any specific network configuration.

The TAS algorithm proposed in this paper is meant as a topology-agnostic information diffusion strategy, thus being a general-purpose solution. For this reason, the FL algorithm, which is topology-agnostic as well, is its natural term of comparison. Both information diffusion strategies are here meant to provide each node with the information needed to locally compute the confidence region as defined by SPS.

The behavior of the algorithms will be illustrated on the toy network represented in Figure 1, where circles represent network nodes and edges between two nodes indicate that they are able to communicate. For each algorithm the evolution of the amount of information available at a node k is described by a table $\mathbf{R}^{(k)}$.

A. Flooding algorithm

FL will be used as a benchmark [2], [25]. When implemented to support the SPS algorithm, *pure* FL works as follows: during the first round, Node k broadcasts its own privy pair $(\boldsymbol{\varphi}_k, y_k)$, and receives data from its neighbors, as dictated by the logical topology. On successive rounds, Node k will also broadcast any previously received pair $(\boldsymbol{\varphi}_i, y_i)$, $i \neq k$ along with its own. In particular, at round r Node k transmits a packet $(\mathbf{t}_r^{(k)}, \mathbf{d}_r^{(k)})$, in which the *tag vector* $\mathbf{t}_r^{(k)}$ indicates the indexes of the nodes whose measurements are present in the packet, whereas the data $\mathbf{d}_r^{(k)}$ contain the measurements and the corresponding regressors $\{(\boldsymbol{\varphi}_j, y_j)\}$, $\forall j \in \mathbf{t}_r^{(k)}$.

Usually, in order to reduce the amount of transmitted information, actual implementations of flooding (*e.g.*, AODV [27]) do not retransmit already transmitted data. In the following we will always refer to such *enhanced* algorithm, that will be simply denoted as flooding.

In this case, Node i is referenced in the tag vector $\mathbf{t}_r^{(k)}$ iff

1) the pair $(\boldsymbol{\varphi}_i, y_i)$ is available at Node k at round $r-1$,
2) the pair $(\boldsymbol{\varphi}_i, y_i)$ has never been broadcast by Node k.

At round $r = 1$, Node k transmits data $\mathbf{d}_1^{(k)}$ consisting of

$$d_{\mathrm{F}} = n_{\mathrm{p}} + 1 \tag{11}$$

real values, corresponding to its measurement and n_{p} regressors. The dimension of data $\mathbf{d}_r^{(k)}$ broadcast by Node k at successive rounds (that is, for $r > 1$) is an integer multiple of d_{F}, possibly zero. The transmission cost d_{TAG} for the tag vector depends on the way it is represented, *e.g.*, as a list of integers, in which case it is of variable length with r, or a constant-size vector of binary flags. The latter is considered in this work. As a consequence, the communication cost of the tag vector is of n binary values per communication round.

Ideally, transmission rounds are repeated until all nodes collect all the information, e.g., by checking the tag vector is full of ones. Upon completion, each node is able to compute (4) and (5), for any $\boldsymbol{\theta}$, and to locally derive the confidence region using the full set of data. In practice, transmission rounds may stop due to information diffusion delay constraints, or when all nodes do not detect any transmitted information from their neighbors over a given time interval.

In the latter cases, the local confidence region characterization may be performed on a reduced, possibly different across nodes, set of data.

Example 1: Table II describes the evolution of the information collected by Node $k = 1$ in the network depicted in Figure 1, when FL is implemented. Before any transmission has taken place, *i.e.*, for $r = 0$, Node 1 only knows its own measurement and regressor, $(\boldsymbol{\varphi}_1, y_1)$.

During the transmission round $r = 1$, Node 1 broadcasts data $\mathbf{d}_1^{(1)} = (\boldsymbol{\varphi}_1, y_1)$. It receives data $\mathbf{d}_1^{(2)} = (\boldsymbol{\varphi}_2, y_2)$ and $\mathbf{d}_1^{(3)} = (\boldsymbol{\varphi}_3, y_3)$ from Nodes 2 and 3 respectively, thus learning measurements and regressors of Nodes 2 and 3.

In round $r = 2$, Node 1 broadcasts $\mathbf{d}_2^{(1)} = \{(\boldsymbol{\varphi}_i, y_i)\}_{i\in\{2,3\}}$. Moreover it receives data generated at Nodes 1, 4, and 7, forwarded by Node 2, (*i.e.*, it receives $\mathbf{d}_2^{(2)} = \{(\boldsymbol{\varphi}_i, y_i)\}_{i\in\{1,4,7\}}$) and the data generated at Nodes 1, 4, and 6, forwarded by Node 3 (*i.e.*, $\mathbf{d}_2^{(3)} = \{(\boldsymbol{\varphi}_i, y_i)\}_{i\in\{1,4,6\}}$). Therefore, at the end of round $r = 2$, Node 1 discovers the measurements of Nodes 4, 6 and 7.

Round	From Node	Data	Tag vector						
0	1	$(\boldsymbol{\varphi}_1, y_1)$	1	0	0	0	0	0	0
1	2	$(\boldsymbol{\varphi}_2, y_2)$	0	1	0	0	0	0	0
	3	$(\boldsymbol{\varphi}_3, y_3)$	0	0	1	0	0	0	0
2	2, 3	$(\boldsymbol{\varphi}_4, y_4)$	0	0	0	1	0	0	0
	3	$(\boldsymbol{\varphi}_6, y_6)$	0	0	0	0	0	1	0
	2	$(\boldsymbol{\varphi}_7, y_7)$	0	0	0	0	0	0	1
3	2, 3	$(\boldsymbol{\varphi}_5, y_5)$	0	0	0	0	1	0	0

TABLE II

TABLE $\mathbf{R}^{(1)}$ OF AVAILABLE INFORMATION AT NODE $k = 1$ WHEN FL IS USED IN THE NETWORK OF FIGURE 1

In round $r = 3$, Node 1 broadcasts $\mathbf{d}_3^{(1)} = \{(\boldsymbol{\varphi}_i, y_i)\}_{i\in\{4,6,7\}}$, and receives data generated at Nodes 3, 5, and 6, forwarded by Node 2, *i.e.*, $\mathbf{d}_3^{(2)} = \{(\boldsymbol{\varphi}_i, y_i)\}_{i\in\{3,5,6\}}$, as well as data from Nodes 2 and 5, forwarded by Node 3, *i.e.*, $\mathbf{d}_3^{(3)} = \{(\boldsymbol{\varphi}_i, y_i)\}_{i\in\{2,5\}}$. Therefore, at the end of round $r = 3$, Node 1 discovers the measurement of Node 5.

If the network is connected, and provided that sufficient transmission rounds are allowed, the FL algorithm diffuses the whole set of data to each node. The computation of the confidence region is accomplished locally using the centralized SPS algorithm. The locally computed confidence regions will be equal only in case there is agreement on the random signs realizations $\{a_{j,k}\}$ used to compute the sign perturbed sums (5), as well as on the random quantities (permutations or random perturbations, [22], [23]) used to resolve ties.

This agreement can be easily accomplished without additional transmission costs by the sharing of the seed of the random generators of the nodes.

B. Tagged and aggregated sums (TAS) algorithm

The TAS algorithm is based on the following consideration. Expanding (4) and (5) one gets,

$$\mathbf{s}_0(\boldsymbol{\theta}) = \mathbf{Q}_n^{-1/2} \left(\sum_{k=1}^{n} \boldsymbol{\varphi}_k y_k - \left(\sum_{k=1}^{n} \boldsymbol{\varphi}_k \boldsymbol{\varphi}_k^T \right) \boldsymbol{\theta} \right) \tag{12}$$

$$\mathbf{s}_j(\boldsymbol{\theta}) = \mathbf{Q}_n^{-1/2} \left(\sum_{k=1}^{n} a_{j,k} \boldsymbol{\varphi}_k y_k - \left(\sum_{k=1}^{n} a_{j,k} \boldsymbol{\varphi}_k \boldsymbol{\varphi}_k^T \right) \boldsymbol{\theta} \right). \tag{13}$$

The evaluation of (12) and (13) for any value of $\boldsymbol{\theta} \in \Theta$ does not necessarily require the

knowledge of each term in the sums but rather of

$$\boldsymbol{\delta}_{1\dots n}=\Bigg\{\underbrace{\sum_{k=1}^{n}\boldsymbol{\varphi}_k y_k}_{n_\mathrm{p}\text{ real values}},\quad\underbrace{\sum_{k=1}^{n}\boldsymbol{\varphi}_k\boldsymbol{\varphi}_k^T}_{n_\mathrm{p}^2\text{ real values}},\quad\underbrace{\left\{\sum_{k=1}^{n}a_{j,k}\boldsymbol{\varphi}_k y_k\right\}_{j=1}^{m-1}}_{n_\mathrm{p}(m-1)\text{ real values}},\quad\underbrace{\left\{\sum_{k=1}^{n}a_{j,k}\boldsymbol{\varphi}_k\boldsymbol{\varphi}_k^T\right\}_{j=1}^{m-1}}_{n_\mathrm{p}^2(m-1)\text{ real values}}\Bigg\} \tag{14}$$

The main idea of the TAS algorithm is to propagate data structures similar to (14), composed of *partial sums* not necessarily ranging from $k=1$ to n, but covering a subset of $\{1,\dots,n\}$. At each transmission round, Node k generates and transmits partial sums built from data previously received from neighbors and stored in $\mathbf{R}^{(k)}$. The main challenge of the TAS algorithm is to determine a way to organize the content of the transmitted partial sums so that each node is able, after the termination of the transmission phase, to build the complete sums (14), or to compute partial sums with the maximum number of elements using the received partial sums. The main advantage of TAS is that the transmitted data sets are of constant size, and do not increase in size with the transmission round as it happens in FL. The size d_{TAS} of the dataset is obtained recalling the amount of data of its components, reported in (14):

$$d_{\mathrm{TAS}}=m\left(n_\mathrm{p}+n_\mathrm{p}\frac{n_\mathrm{p}+1}{2}\right) \tag{15}$$

The evaluation of d_{TAS} takes into account the fact that $\boldsymbol{\varphi}_k\boldsymbol{\varphi}_k^T$ is symmetric[4]. Note that the size of the dataset is fixed, *independently* of the number of elements in the partial sums. As in FL, the tag vector has to be transmitted along with the data set at each transmission round. Notice that, with the representation chosen in this work, the transmission cost of the tag vector in the FL and TAS algorithms is the same.

The TAS algorithm, whose structure is reported in Algorithm 1, consists of six phases, namely, i) initialization, ii) reception, iii) distillation, iv) aggregation, v) transmission, and vi) wrap-up. The detailed description of each phase is reported hereafter, while the corresponding pseudo codes are in Appendix A.

[4]Since $\sum_{k=1}^{n}\boldsymbol{\varphi}_k\boldsymbol{\varphi}_k^T$ is symmetric, instead of transmitting all its $\boldsymbol{n_\mathrm{p}}^2$ elements, it is sufficient to transmit $\boldsymbol{n_\mathrm{p}}$ values for the diagonal plus $\sum_{\boldsymbol{d=1}}^{\boldsymbol{n_\mathrm{p}-1}}\boldsymbol{d}=\frac{\boldsymbol{n_\mathrm{p}(n_\mathrm{p}-1)}}{\boldsymbol{2}}$ values for the upper (or lower) part, that gives $n_\mathrm{p}\frac{n_\mathrm{p}+1}{2}$. The same holds for the $(m-1)$ terms $\left\{\sum_{k=1}^{n}a_{j,k}\boldsymbol{\varphi}_k\boldsymbol{\varphi}_k^T\right\}_{j=1}^{m-1}$.

Algorithm 1 TAS algorithm

```
Initialization
for r = 1 to MaxRound do
    Reception
    Distillation
    Aggregation
    Transmission
end for
Wrap-up
```

i) *Initialization phase*, see Algorithm 2. As in the FL protocol, the transmitted packet is formed by a data set and by a tag vector. During the initialization phase, Node k, $\forall k \in \{1,\dots,n\}$ creates the packet $\left(\mathbf{t}_1^{(k)}, \mathbf{d}_1^{(k)}\right)$ to be sent in round $r=1$. The tag vector $\mathbf{t}_1^{(k)}$ flags only Node k.

$$\mathbf{t}_1^{(k)} = [0, \quad \dots, \quad \underset{\dots}{0}, \quad \underset{\text{k-1}}{1}, \quad \underset{\text{k}}{0}, \quad \underset{\text{k+1}}{\dots}, \quad 0] \tag{16}$$

The data set $\mathbf{d}_1^{(k)}$ contains the local quantities related to Node k

$$\mathbf{d}_1^{(k)} = \left\{ \boldsymbol{\varphi}_k y_k, \left\{ \boldsymbol{\varphi}_k \boldsymbol{\varphi}_k^T \right\}, \left\{ a_{j,k} \boldsymbol{\varphi}_k y_k \right\}_{\forall j}, \left\{ a_{j,k} \boldsymbol{\varphi}_k \boldsymbol{\varphi}_k^T \right\}_{\forall j} \right\}. \tag{17}$$

After initialization, the reception, distillation, aggregation, and transmission phases are sequentially repeated until a termination condition is met (*e.g.*, until a given number of rounds have been completed, as in Algorithm 1).

ii) *Reception phase*, see Algorithm 3. At each round r, Node k collects the messages containing the partial sums transmitted by its neighbors (according to the given logical topology), whose set is denoted $\mathcal{N}(k)$.

iii) *Distillation phase*, see Algorithm 4. At the end of the reception phase of round r, Node k compares the incoming tag vectors $\mathbf{t}_r^{(j)}$, $j \in \mathcal{N}(k)$ to the previously received tag vectors, to detect whether the packets received at round r contain new information. If it appears that a part of the data referenced in $\mathbf{t}_r^{(j)}$ have been previously received, these redundant data are removed from the corresponding partial sum and $\mathbf{t}_r^{(j)}$ is updated accordingly, see Lines 3 to 6. The resulting partial sums are then stored in $\mathbf{R}^{(k)}$. The same procedure is applied to already stored partial sums, see Lines 7 to 9. This phase reduces the number of contributors to each partial sum, so that the different partial sums can be more easily recombined, in the following *aggregation phase*, with each contributor counted no more than once.

Round	From Node	Data	Tag vector						
0	1	$\boldsymbol{\delta}_1$	1	0	0	0	0	0	0
1	2	$\boldsymbol{\delta}_2$	0	1	0	0	0	0	0
	3	$\boldsymbol{\delta}_3$	0	0	1	0	0	0	0
2	2	$\boldsymbol{\delta}_{4,7}$	C	0	0	1	0	0	1
	3	$\boldsymbol{\delta}_{4,6}$	C	0	0	1	0	1	0
3	2	$\boldsymbol{\delta}_{5,6}$	0	0	C	0	1	1	0
	3	$\boldsymbol{\delta}_5$	0	C	0	0	1	0	0

TABLE III

TABLE $\mathbf{R}^{(1)}$ FOR NODE $k=1$ USING TAS IN THE NETWORK OF FIGURE 1; C INDICATES ELEMENTS THAT HAVE BEEN REMOVED FROM THE TAG VECTOR AND PARTIAL SUMS DURING THE DISTILLATION PHASE

Example 2 (Distillation phase): Consider again the network of Figure 1 and the evolution of $\mathbf{R}^{(1)}$ given in Table III. As in FL, for $r=0$, Node 1 only holds its own data and forms partial sums from these data stored in

$$\boldsymbol{\delta}_1=\left\{\boldsymbol{\varphi}_1 y_1, \left\{\boldsymbol{\varphi}_1\boldsymbol{\varphi}_1^T\right\}, \left\{a_{j,1}\boldsymbol{\varphi}_1 y_1\right\}_{\forall j}, \left\{a_{j,1}\boldsymbol{\varphi}_1\boldsymbol{\varphi}_1^T\right\}_{\forall j}\right\}.$$

During round $r=1$, Node 1 broadcasts these partial sums and receives partial sums formed with the privy data from Node 2 and partial sums formed with the privy data from Node 3. During round $r=2$, Node 1 receives a packet containing partial sums combining data from Nodes 1, 4, and 7, forwarded by Node 2, as well as a packet containing partial sums combining data from Nodes 1, 4, and 6, forwarded by Node 3. The content of these two packets is stored in $\mathbf{R}^{(1)}$, after having removed the contribution related to Node 1 from each previously received partial sum (this is indicated by a C in the tag vector in Table III). Node 1 thus gets

$$\begin{aligned}\boldsymbol{\delta}_{4,6} = \Bigg\{ &\sum_{k\in\{4,6\}} \boldsymbol{\varphi}_k y_k, \sum_{k\in\{4,6\}} \boldsymbol{\varphi}_k\boldsymbol{\varphi}_k^T, \\ &\left\{\sum_{k\in\{4,6\}} a_{j,k}\boldsymbol{\varphi}_k y_k\right\}_{\forall j}, \left\{\sum_{k\in\{4,6\}} a_{j,k}\boldsymbol{\varphi}_k\boldsymbol{\varphi}_k^T\right\}_{\forall j}\Bigg\}\end{aligned} \tag{18}$$

and $\boldsymbol{\delta}_{4,7}$. At the end of round $r=3$, Node 1 receives a packet with partial sums combining data from Nodes 3, 5, and 6, forwarded by Node 2, as well as a packet with partial sums combining data from Nodes 2 and 5, forwarded by Node 3.

iv) *Aggregation phase*, see Algorithm 5. To create the packet to be broadcast at round r, Node k aggregates the partial sums available in $\mathbf{R}^{(k)}$ at round $r-1$ and which were *not* previously aggregated. This is done by summing the available partial sums to produce $\mathbf{d}_r^{(k)}$

Round	From Node	Data	Tag vector						
0	2	$\boldsymbol{\delta}_2$	0	1	0	0	0	0	0
1	1	$\boldsymbol{\delta}_1$	1	0	0	0	0	0	0
	4	$\boldsymbol{\delta}_4$	0	0	0	1	0	0	0
	7	$\boldsymbol{\delta}_7$	0	0	0	0	0	0	1
2	1	$\boldsymbol{\delta}_3$	0	C	1	0	0	0	0
	4	$\boldsymbol{\delta}_6$	0	C	C	0	0	1	0
	7	$\boldsymbol{\delta}_5$	0	C	0	0	1	0	0

TABLE IV

TABLE $\mathbf{R}^{(2)}$ FOR NODE $k = 2$ USING TAS IN THE NETWORK OF FIGURE 1; C INDICATES ELEMENTS THAT HAVE BEEN REMOVED FROM THE TAG VECTOR AND PARTIAL SUMS DURING THE DISTILLATION PHASE

and merging the related tag vectors to produce $\mathbf{t}_r^{(k)}$. In order to avoid duplication of terms in the sums, rows i and j of $\mathbf{R}^{(k)}$ can be merged in $\left(\mathbf{t}_r^{(k)}, \mathbf{d}_r^{(k)}\right)$ iff the intersection of i-th and j-th row tag vectors is empty. If this condition is not met, only the row with smallest index is aggregated in a transmitted packet.

Example 3 (Aggregation phase): Consider the evolution of $\mathbf{R}^{(2)}$ for Node 2 given in Table IV. At the end of round $r = 1$, Node 2 holds partial sums related to the data from Nodes 1, 2, 4, and 7, stored in $\boldsymbol{\delta}_1$, $\boldsymbol{\delta}_2$, $\boldsymbol{\delta}_4$, and $\boldsymbol{\delta}_7$. A packet containing $\boldsymbol{\delta}_2$ has already been transmitted in round $r = 1$. The other tag vectors do not intersect, as a consequence, the aggregated sums will involve $\boldsymbol{\delta}_1$, $\boldsymbol{\delta}_4$, and $\boldsymbol{\delta}_7$.

The distillation phase facilitates the aggregation and wrap-up phases. Moreover, it allows to get sparser tag vectors, which may then be more efficiently combined.

v) *Transmission phase*, see Algorithm 6. The message obtained at the end of the aggregation phase is broadcast to all neighbor nodes. After the last transmission phase, the objective for Node k is the computation of the local confidence region, using the data collected so far and aggregated in the final partial sum $\boldsymbol{\delta}_F^{(k)}$, evaluated in the wrap-up phase. The information diffusion process stops for Node k when it has collected all the information from other nodes or, more realistically, when a certain time has expired.

vi) *Wrap-up phase*, see Algorithm 7. The wrap-up phase can be performed by a node whenever it needs to compute the confidence region during or at the end of the information diffusion process. For that purpose, Node k evaluates a linearly weighted sum $\boldsymbol{\delta}_F^{(k)} = \sum_l \widehat{b}_l^{(k)} \boldsymbol{\delta}_l^{(k)}$, where $\boldsymbol{\delta}_l^{(k)}$ contains the partial sums at the l-th row of $\mathbf{R}^{(k)}$ and $\widehat{\mathbf{b}}^{(k)}$ is a vector of weights. The non-zero entries of $\widehat{\mathbf{b}}^{(k)}$ select the rows of $\mathbf{R}^{(k)}$ to be combined in the partial sums.

To obtain $\widehat{\mathbf{b}}^{(k)}$, consider the tag matrix $\mathbf{T}^{(k)}$ of $\mathbf{R}^{(k)}$, with elements $t_{l,i}^{(k)}$, with l and i

denoting the row and column indexes, respectively. If $\mathbf{T}^{(k)}$ is of full rank n, then $\mathbf{R}^{(k)}$ contains a contribution from all nodes of the network and as in network coding, one may isolate each individual contribution via Gaussian elimination performed on $\mathbf{T}^{(k)}$ and proceed at the considered node in the same way as for the centralized SPS.

A second case is when $n^{(k)}$ columns of $\mathbf{T}^{(k)}$ contain 1s and the rank of $\mathbf{T}^{(k)}$ is also equal to $n^{(k)}$. In this case, only $n^{(k)}$ nodes have contributed to the partial sums stored in the rows of $\mathbf{R}^{(k)}$. Since $\mathbf{T}^{(k)}$ is of rank $n^{(k)}$, it is again possible to recover via Gaussian elimination the individual contributions of a subset $\mathcal{I}$ of $n^{(k)}$ out of the n nodes. Provided that $n^{(k)} \geqslant n'$, $\mathbf{Q}_{\mathcal{I}}$ will be invertible and one will be able to obtain a LS estimate and its corresponding confidence region from a subset of $n^{(k)}$ data. When $n^{(k)} < n'$, more rounds have to be performed.

The last case to be considered is when $n^{(k)}$ columns of $\mathbf{T}^{(k)}$ contain 1s and the rank of $\mathbf{T}^{(k)}$ is strictly less than $n^{(k)}$. In that case, one may try to search the solution of the following constrained optimization problem

$$\widehat{\mathbf{b}}^{(k)} = \arg\max_{\mathbf{b}} \ \mathbf{b}^T \mathbf{T}^{(k)} \mathbf{1}, \tag{19}$$

with the constraints

$$c_i^{(k)} = \sum_l \ b_l \ t_{l,i}^{(k)} \in \{0,1\}, \quad i = 1, 2, \ldots, n. \tag{20}$$

$$\det \sum_l b_l \left(\sum_{k \in \mathbf{t}_l^{(k)}} \boldsymbol{\varphi}_k \boldsymbol{\varphi}_k^T \right) \neq 0. \tag{21}$$

The constraints (20) are related to the presence indicator of the quantities associated to Nodes $i = 1, \ldots, n$. Imposing $c_i^{(k)} \in \{0,1\}$ in (20) ensures that all measurements contribute similarly to the final sign perturbed sums, with some measurements possibly not contributing at all. In the latter case, one obtains a confidence region associated to the LS estimate of θ^* involving only the corresponding subset of sensor measurements. The constraint (21) is introduced to allow the computation of an approximation of $\mathbf{Q}_n^{-1/2}$ relying on possibly less than n terms.

The constrained integer programming problem (19)-(21) is NP-hard in general. If the constraint (21) is verified only *a posteriori*, one gets a linear cost function and (20) can be formulated as quadratic equality constraints. A further relaxation of (20) can be considered imposing only that $c_i^{(k)} \in [0,1]$. One gets then a linear programming problem, easier to solve, but that may provide a solution quite far from that of the original integer programming

problem. More precisely, if for the solution, $c_i^{(k)} \in]0,1[$, the i-th measurement will not contribute with a unit weight. One obtains at the best a weighted LS estimate of $\boldsymbol{\theta}^*$ and its associated confidence region, and not the original LS estimate from equally-weighted data.

An alternative sub-optimal wrap-up algorithm is provided in Appendix A, which is less energy demanding owing to the lower computational effort required. The idea is closely related to that of the aggregation algorithm. The main difference is that in the wrap-up algorithm, the rows of $\mathbf{R}^{(k)}$ are first sorted by decreasing order of the weight of the rows of the tag matrix $\mathbf{T}^{(k)}$. The idea is to start aggregating partial sums starting with the partial sums to which a maximum number of nodes have contributed. The gap to optimality of this heuristic algorithm can be upper-bounded by considering the number of column $n^{(k)}$ of $\mathbf{T}^{(k)}$ containing 1s. Since $n^{(k)}$ represents the number of different nodes that have contributed to one of the partial sums stored in $\mathbf{R}^{(k)}$, the optimal wrap up performed solving (19)-(21) cannot aggregate data from more than $n^{(k)}$ nodes. The gap to optimality is thus less than the difference between $n^{(k)}$ and the number of aggregated data from different nodes, *i.e.*, the number of 1s in the final aggregated tag vector.

In any case, before starting the final wrap-up, a node should have a matrix $\mathbf{T}^{(k)}$ such that $n^{(k)} \geqslant n'$ to have a chance wrapping-up data from enough nodes to get an invertible matrix $\mathbf{Q}_{\mathcal{I}}$.

Once a satisfying solution has been found, Node k can locally compute an exact confidence region based on $\boldsymbol{\delta}_F^{(k)}$, from which the following quantities are evaluated

$$\widetilde{\mathbf{s}}_0^{(k)}(\boldsymbol{\theta}) = \widetilde{\mathbf{Q}}^{-1/2} \sum_{i=1}^{n} c_i^{(k)} \boldsymbol{\varphi}_i \left(y_i - \boldsymbol{\varphi}_i^T \boldsymbol{\theta}\right) \tag{22}$$

$$\widetilde{\mathbf{s}}_j^{(k)}(\boldsymbol{\theta}) = \widetilde{\mathbf{Q}}^{-1/2} \sum_{i=1}^{n} c_i^{(k)} a_{j,i} \boldsymbol{\varphi}_i \left(y_i - \boldsymbol{\varphi}_i^T \boldsymbol{\theta}\right) \ \forall j = 1, \ldots, m-1, \tag{23}$$

with

$$\widetilde{\mathbf{Q}} = \frac{1}{\sum_{i=1}^{n} c_i^{(k)}} \sum_{i=1}^{n} c_i^{(k)} \boldsymbol{\varphi}_i \boldsymbol{\varphi}_i^T. \tag{24}$$

Various confidence regions may then be defined and evaluated from (22) and (23).

If several satisfying solutions for (19-20) have been found, the one maximizing (21) should be selected to get the smallest confidence region, as in D-optimal experiment design [28].

Remark 1: The TAS algorithm is inspired from network coding [29], [30]. The main difference is that Node k does not need to recover the privy data of all nodes, but the decoding of their partial sums suffices.

Remark 2: The efficiency of TAS with respect to FL comes from the fact that the size d_{TAS} of the data sets exchanged does not increase as the number of rounds increases, as it happens in FL.

C. Consensus algorithm

Given that the SPS algorithm does not require the single terms appearing in (12) and (13) but rather their sum, a possibility to compute (12) and (13) in a distributed way, is to launch an average consensus algorithm [31]–[34], converging to (14), as proposed in [1]. For this information diffusion strategy, $\mathbf{R}^{(k)}$ is always composed of a single row, storing the consensus state vector. Further details can be found in [1], [31]–[34]. Consensus algorithms will be considered in the numerical results section, anyway we will not put more emphasis since they showed a poor performance in terms of generated traffic load and convergence speed, as investigated in [1].

IV. Traffic load on various network topologies

In this section, the amount of transmitted data for distributed confidence region characterization is analyzed for both FL and TAS. Their performances are compared on different *logical* topologies, with particular reference to generic trees, that is trees with an arbitrary number of children for each node (Section IV-A), binary trees (Section IV-B) and clustered networks (Section IV-C), that are the most commonly used topologies in practical applications [4]. Section V considers also completely unstructured networks.

Remind that d_{F}, given by (11), denotes the numbers of real-valued scalars (possibly quantized) that a single data (measurement and vector of regressors) is composed of when the FL algorithm is used. With the FL algorithm, a packet usually contains several data, and thus an integer multiple of d_{F} scalars. Similarly, d_{TAS}, given by (15), is the fixed amount of (possibly quantized) real-valued scalars that are carried by a packet transmitted by a given node when considering the TAS algorithm.

The transmission cost of the tag vector, consisting of n binary values, is the same across transmission rounds, and whatever the information diffusion strategy.

A. Tree Topology

The tree topology is one of the most common logical topology encountered in WSNs. It might be the consequence of a particular physical deployment of nodes or the result of a spanning tree routing procedure.

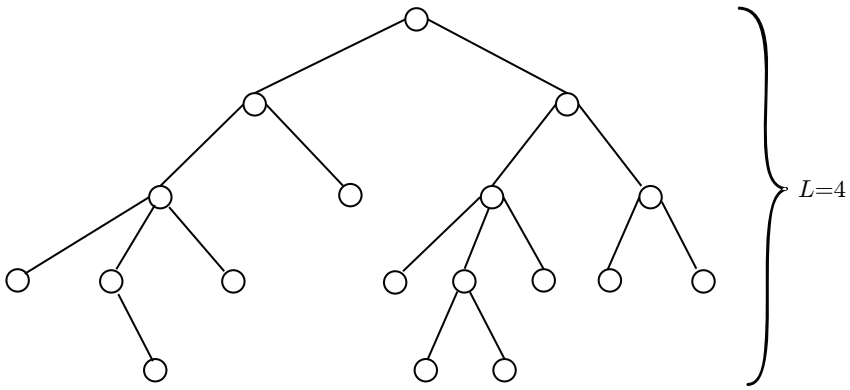

Fig. 2. Generic tree topology with $L = 4$, where $\lambda(0) = 1$, $\lambda(1) = 2$, $\lambda(2) = 4$, $\bar{\lambda}(2) = 1$, $\lambda(3) = 8$, $\bar{\lambda}(3) = 6$, $\lambda(4) = \bar{\lambda}(4) = 3$.

Usually, tree topologies resulting from routing algorithms specifically designed for WSNs introduce some constraints in the way data travel, according to energy saving strategies. For instance, only nodes at a single level of the tree may be allowed to transmit during each round and nodes belonging to that level can communicate only with nodes belonging to the successive level [35], as all the other nodes are in *sleep state*. For this reason, the generic tree topology addressed in this section will be investigated assuming that a message broadcast by a node in the forward phase is only exploited by its parent. This hypothesis will be removed in Section IV-B, addressing the particular case of binary trees, that discusses also what happens when children nodes can overhear transmissions carried out by their parents.

Consider now a generic tree topology, *i.e.*, a tree where each node has an arbitrary, yet known, number of children, possibly zero. Denote with $\lambda(\ell)$ the number of nodes at Level ℓ and with $\overline{\lambda}(\ell)$ the number of nodes at Level ℓ that have no children, with ℓ ranging from $\ell = 0$ (the root) to $\ell = L$ (the leaves). Of course $\lambda(0) = 1$, since the tree is single rooted. The total number of nodes forming the network is therefore $n = \sum_{\ell=0}^{L} \lambda(\ell)$. An example of these networks is depicted in Figure 2.

1) FL algorithm: The amount of data that needs to be transmitted in the forward phase from Level L to Level $L-1$ is $f_{L,L-1} = \lambda(L)\, d_{\mathrm{F}}$. When $1 \leqslant \ell < L$, this amount, from Level ℓ to Level $\ell - 1$, is $f_{\ell,\ell-1} = (\lambda(L) + \cdots + \lambda(\ell))\, d_{\mathrm{F}}$. In the backward phase, the amount of data that needs to be transmitted from Level 0 to Level 1 is $b_{0,1} = n d_{\mathrm{F}}$. When $1 \leqslant \ell < L$, from Level ℓ to Level $\ell + 1$, it is $b_{\ell,\ell+1} = \left(\lambda(\ell) - \overline{\lambda}(\ell)\right) n\, d_{\mathrm{F}}$.

Finally, the amount of data that has to be transmitted with the FL algorithm to share all

data between nodes in the network is

$$\begin{aligned} n_{\text{FL}}^{\text{GT}} =& \left(\lambda\left(L\right) + \left(\lambda\left(L\right) + \lambda\left(L-1\right)\right) + \cdots + \sum_{\ell=1}^{L} \lambda\left(\ell\right) \right) d_{\text{F}} \\ & + n\, d_{\text{F}} + \sum_{\ell=1}^{L-1} \left(\lambda\left(\ell\right) - \overline{\lambda}\left(\ell\right) \right) n d_{\text{F}} \\ =& L n\, d_{\text{F}} - \left(\lambda\left(0\right) + \left(\lambda\left(0\right) + \lambda\left(1\right)\right) + \cdots + \sum_{\ell=0}^{L-1} \lambda\left(\ell\right) \right) d_{\text{F}} \\ & + n^2 d_{\text{F}} - \lambda\left(L\right) n\, d_{\text{F}} - \left(\sum_{\ell=0}^{L-1} \overline{\lambda}\left(\ell\right) \right) n\, d_{\text{F}}. \end{aligned} \tag{25}$$

2) TAS algorithm: In the forward phase, the TAS distillation and aggregation phases take place after each transmission round. The data reaching the root corresponds to the elements required to evaluate the unperturbed and perturbed sums that would be obtained in a centralized version of the algorithm. This way of operating ensures thus an exact retrieval of the entire sums (4) and (5). In the backward phase, this information is spread over the tree without any further processing. As already mentioned, all data packets have a constant size d_{TAS}.

The amount of data to be transmitted in the forward direction from Level ℓ to Level $\ell - 1$ is $\lambda\left(\ell\right) d_{\text{TAS}}$. In the backward direction, from Level ℓ to Level $\ell + 1$, it is $\left(\lambda\left(\ell\right) - \overline{\lambda}\left(\ell\right) \right) d_{\text{TAS}}$, since nodes without children do not transmit further. Accounting for both phases, one gets

$$\begin{aligned} n_{\text{TAS}}^{\text{GT}} =& \left(\sum_{\ell=1}^{L} \lambda\left(\ell\right) \right) d_{\text{TAS}} + \sum_{\ell=0}^{L-1} \left(\lambda\left(\ell\right) - \overline{\lambda}\left(\ell\right) \right) d_{\text{TAS}} \\ =& (2n-1) d_{\text{TAS}} - \lambda(L) d_{\text{TAS}} - \left(\sum_{\ell=0}^{L-1} \overline{\lambda}\left(\ell\right) \right) d_{\text{TAS}}. \end{aligned} \tag{26}$$

Starting from the general expressions (25) and (26), in Section IV-B we investigate the amount of data transmitted by FL and TAS in the significant case of binary trees.

B. Binary Tree Topology

Consider a *single-rooted complete binary tree* with $L + 1$ levels. In this case,

$$\lambda\left(\ell\right) = 2^{\ell}\,, \tag{27}$$

$$\overline{\lambda}\left(\ell\right) = 0 \quad \text{for } \ell = 0, 1, ..., L-1\,, \tag{28}$$

$$n = \sum_{\ell=0}^{L} \lambda\left(\ell\right) = 2^{L+1} - 1\,. \tag{29}$$

1) FL algorithm: Using (27), (28), and (29) in (25), the amount of data transmitted by FL in a generic tree can be specialized for the binary tree case, to get

$$\begin{aligned} n_{\text{FL}}^{\text{BT}} &= \left(\frac{(n+1)^2}{2} + \left(\log_2 (n+1) - \frac{7}{2} \right)(n+1) + 3 \right) d_{\text{F}} \\ &\simeq \frac{(n+1)^2}{2} d_{\text{F}}. \end{aligned} \tag{30}$$

for n sufficiently large.

If we remove the hypothesis that nodes enter in a *sleep state* at the end of their transmission round (thus allowing bidirectional communications), it is true that a message transmitted by a node in the forward phase can be processed also by its children. This property can be used in the backward phase by FL (denoted in this case FL-B) to reduce the amount of data to propagate. In this case (25) boils down to

$$\begin{aligned} n_{\text{FL-B}}^{\text{BT}} &= \left((n+1) \frac{1}{2} (n-1) + 1 \right) d_{\text{F}} \\ &= \frac{(n+1)^2}{2} d_{\text{F}} - n\, d_{\text{F}}. \end{aligned} \tag{31}$$

One observes that $n_{\text{FL}}^{\text{BT}} > n_{\text{FL-B}}^{\text{BT}}$. As expected, accounting for data overheard by children in the forward phase reduces the amount of data to be transmitted. For large networks, however, both (30) and (31) scale quadratically in n, thus making the bidirectional tree not convenient, as it is more power consuming.

2) TAS algorithm: The amount of data transmitted by TAS in the binary tree case can be derived using (27), (28), and (29) in (26), thus obtaining

$$n_{\text{TAS}}^{\text{BT}} = \frac{3}{2} (n-1)\, d_{\text{TAS}}. \tag{32}$$

With the TAS algorithm $n_{\text{TAS}}^{\text{BT}}$ scales thus linearly with n.

3) Comparison: When comparing (30), (31), and (32), asymptotically, the TAS algorithm is the most efficient, since the amount of data to be exchanged on the network scales linearly with the number of nodes n, where it scales in n^2 with the other algorithms. Nevertheless, for small values of n, the fact that $d_{\text{TAS}} > d_{\text{F}}$ can make the TAS algorithm less efficient.

On a binary tree, TAS is more efficient than FL-B when

$$(3n-3) d_{\text{TAS}} < (n^2+1) d_{\text{F}}.$$

Using (15) and (11) one obtains the following condition

$$\left(n^2+1\right) K_1 - 3n + 3 > 0, \tag{33}$$

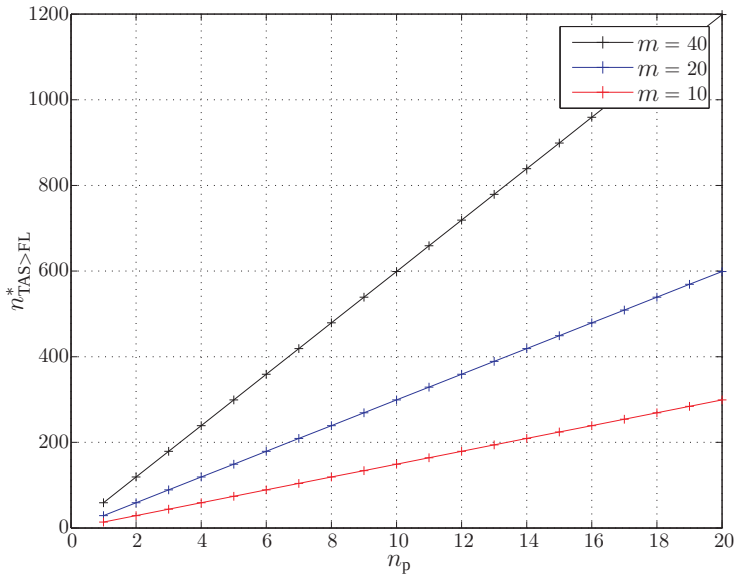

Fig. 3. Critical value $n^*_{\mathrm{TAS>FL}}$, as a function of n_{p}, on binary trees, for several values of m.

where $K_1 = \frac{n_p+1}{\left(n_p+n_p\frac{n_p+1}{2}\right)m}$. For sufficiently large n, (33) is always satisfied, for all n_p and m. Moreover, when n is larger than

$$n^*_{\mathrm{TAS>FL}} = \frac{3+\sqrt{9-4K_1\left(3+K_1\right)}}{2K_1}, \tag{34}$$

TAS is more efficient than FL. Figure 3 represents $n^*_{\mathrm{TAS>FL}}$ as a function of n_p, considering $m = 10$, $m = 20$, and $m = 40$. The behaviour is not exactly linear, but when n_p grows large, $K_1 \approx \frac{2}{n_p m}$ and $n^*_{\mathrm{TAS>FL}} \approx \frac{3}{2}n_p m$.

C. Clustered Topology

Consider a clustered network, formed by n nodes, structured on a single level of hierarchy, as depicted in Figure 4. The network is hence assumed to be divided in n_{c} clusters. The i-th cluster comprises a random number of nodes n^{c}_i, including the clusterhead, that is the special node responsible for aggregating the local data of its sons. The subnetwork formed by clusterheads is considered to be fully connected: Clusterheads can directly communicate with each other. Moreover, each node in a cluster is assumed to directly communicate with its clusterhead (and vice-versa).

1) FL algorithm: All nodes in a cluster can overhear broadcast transmissions operated by the corresponding clusterhead. Therefore, the amount of data to be transmitted when

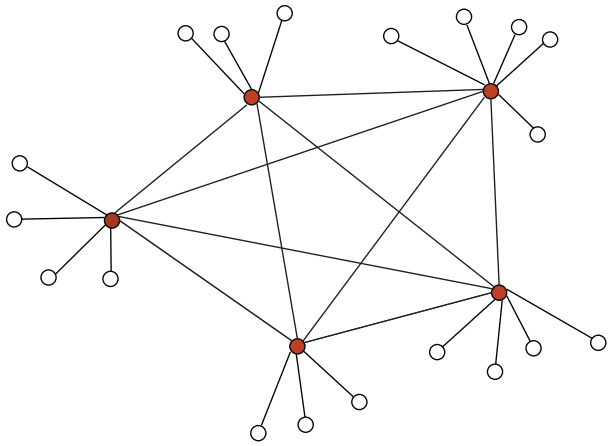

Fig. 4. A clustered topology. Clusterheads are indicated in red.

employing the FL algorithm is

$$\begin{aligned} n_{\text{FL}}^{\text{CN}} &= \left((n - n_{\text{c}}) + n + (n_{\text{c}} - 1)\, n\right) d_{\text{F}} \\ &= \left(n - n_{\text{c}} + n_{\text{c}} n\right) d_{\text{F}}. \end{aligned} \tag{35}$$

This is because all nodes, apart from clusterheads, initially transmit their local information to clusterheads, leading to $(n - n_{\text{c}})d_{\text{F}}$ transmitted scalar data. Then clusterheads broadcast the received data and their own, this forming a total flow of nd_{F} scalar data. At this point, all nodes in each cluster are completely informed about data related to their respective cluster. Finally, there is a backward transmission during which each clusterhead is transmitting towards its cluster all the nd_{F} scalar data except the ones that it previously transmitted, leading to further $(n_{\text{c}} - 1)\, nd_{\text{F}}$ transmitted scalars, composed of n_{c} clusterheads transmitting not n, but $(n - n_i^{\text{c}})d_{\text{F}}$ scalar data, *i.e.*, a total of $\sum_{i=1}^{n_{\text{c}}} (n - n_i^{\text{c}})\, d_{\text{F}} = (n_{\text{c}} - 1)\, nd_{\text{F}}$.

2) TAS algorithm: On this topology, the TAS algorithm transmission phases can be organized as follows. At the beginning, each node, with the exception of clusterheads, transmits the partial sums calculated with its own data, corresponding to d_{TAS} real values per node. Then each clusterhead aggregates the local data of all nodes in its cluster. Successively, clusterheads transmit to all other clusterheads their aggregated data. Since the network of clusterheads is fully connected, a single broadcast transmission for each of the clusterheads suffices for all clusterheads being capable to construct the completely aggregated data. The amount of scalar data, that has to be transmitted, is thus

$$n_{\text{TAS}}^{\text{CN}} = \left((n - n_{\text{c}}) + n_{\text{c}} + n_{\text{c}}\right) d_{\text{TAS}} = \left(n + n_{\text{c}}\right) d_{\text{TAS}}.$$

This accounts for the initial $n - n_{\mathrm{c}}$ transmissions and the subsequent actions of clusterheads, that should broadcast to each other the partially aggregated data and then broadcast, towards nodes forming their cluster, the completely aggregated data.

3) Comparison: TAS is better than FL when $n_{\mathrm{TAS}}^{\mathrm{CN}} < n_{\mathrm{FL}}^{\mathrm{CN}}$, *i.e.*, when

$$\begin{aligned} (n - n_{\mathrm{c}} + n_{\mathrm{c}} n)\, d_{\mathrm{F}} - (n + n_{\mathrm{c}})\, d_{\mathrm{TAS}} > 0 \\ \left(1 + \frac{n_{\mathrm{c}}(n-2)}{n + n_{\mathrm{c}}}\right) \frac{d_{\mathrm{F}}}{d_{\mathrm{TAS}}} > 1. \end{aligned} \tag{36}$$

With n sufficiently large, one has

$$\left(1 + \frac{n_{\mathrm{c}}(n-2)}{n + n_{\mathrm{c}}}\right) \frac{d_{\mathrm{F}}}{d_{\mathrm{TAS}}} \approx (n_{\mathrm{c}} + 1) \frac{d_{\mathrm{F}}}{d_{\mathrm{TAS}}}.$$

This implies that TAS is better than FL when

$$n_{\mathrm{c}} > \frac{d_{\mathrm{TAS}}}{d_{\mathrm{F}}} - 1.$$

Remark 3: In Section III we indicated that the TAS algorithm proposed in this paper is meant as a topology-agnostic information diffusion strategy. Of course, given the network topology, specialized information diffusion strategies can be designed, likely providing better performance. For instance, in the case of the clustered topology here considered, one could imagine a mixed FL+TAS approach in which, during the first transmission phase, each node of a cluster conveys to the clusterheads d_{F} data, composed by its privy data with no aggregation (as done by FL). Then, the tagged and aggregated sums are evaluated by the clusterheads, that make data circulate as dictated by TAS. In this case, the amount of scalar data that has to be transmitted is

$$n_{\mathrm{FL+TAS}}^{\mathrm{CN}} = (n - n_{\mathrm{c}})\, d_{\mathrm{F}} + 2 n_{\mathrm{c}} d_{\mathrm{TAS}},$$

which is always lower than $n_{\mathrm{TAS}}^{\mathrm{CN}}$. Morevover one has $n_{\mathrm{FL+TAS}}^{\mathrm{CN}} < n_{\mathrm{FL}}^{\mathrm{CN}}$ as soon as $n > 2\frac{d_{\mathrm{TAS}}}{d_{\mathrm{F}}}$.

V. Simulation Results

In this section, all simulations results have been obtained considering sensor nodes randomly deployed over a square of side of one measurement unit, which transmit information over lossless links (*i.e.*, no transmission errors and no packet collisions), while confidence regions have been evaluated with the interval analysis techniques described in [24] and the Intlab library [36] for interval computations. Data are generated considering the model (1), with randomly generated parameters and regressors using realizations of independent zero-mean unit variance Gaussian variables. The noise corrupting data is also zero-mean Gaussian, with a variance adjusted to get a signal-to-noise ratio of 15 dB.

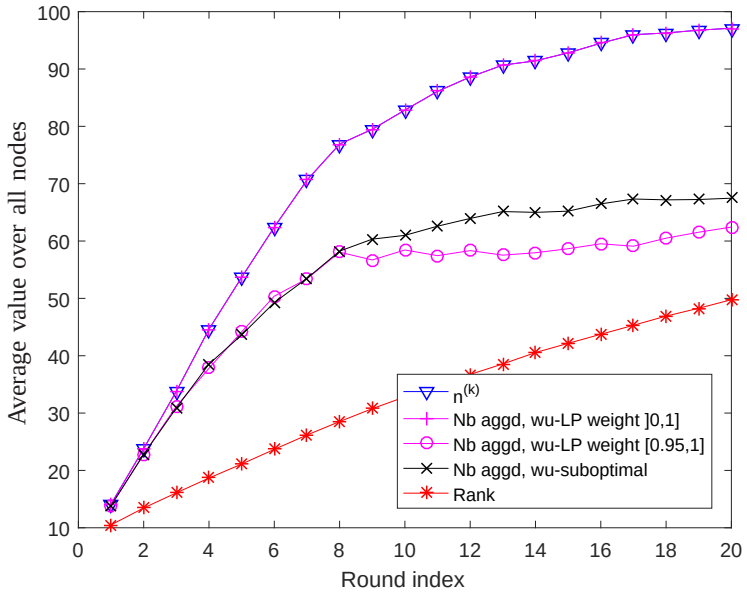

Fig. 5. Behavior of the TAS algorithm with a random unstructured topology, as a function of the round index.

A. Behavior of the TAS algorithm

One considers here a random unstructured topology to see how information propagates within the network with the TAS algorithm. Figure 5 describes the evolution as a function of the number of rounds of the average rank of the tag matrices, the average number of data wrapped-up with the suboptimal wrap-up described in Algorithm 7, and that obtained using linear programming, see Section III-B. With the latter approach, two plots are reported, one is showing the average number of data contributing to the final sum with a weight within the interval $[0.95, 1]$, the second is the average number of data contributing, whatever their strictly positive weight. Finally, the average value of $n^{(k)}$ is provided. Averages are taken over all nodes. For the considered simulation, a network of $n = 100$ nodes is investigated. The corresponding graph is connected with an average node connectivity of 6.38 and a diameter of 13.

One observes first that the average rank increases slower than $n^{(k)}$. The sum of the contributions of all nodes may thus be obtained before obtaining each individual contribution. Second, the wrap-up via linear programming is able to collect most of the data, even if their weight is not necessarily one in the final sum. The suboptimal wrap-up algorithm performs somewhat worse than the wrap-up via linear programming, but is able to gather an amount of data close to that contributing with a weight close to 1 in the wrap-up using linear programming. Moreover, all these quantities increase fast in the first rounds and slower

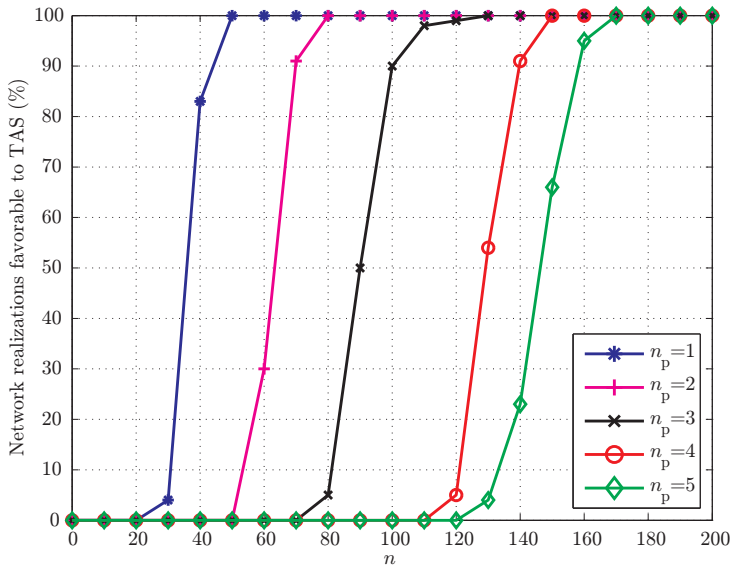

Fig. 6. Percentage of network realizations favorable to TAS, in terms of required data exchanges, compared to FL, as a function of the number of nodes forming a random tree topology for different values of n_{p}. 100 random tree realizations are considered for each value of n.

after several rounds. This is due to the fact that at the beginning, each packet contains new information, whereas packets in the last rounds contain only limited new information. Moreover, the aggregation phase has more difficulties to aggregate tag vectors received in the last rounds, which contains already many contributions from different nodes and are likely to contain at least partly similar contributions. When the network is more structured, this phenomenon does not appear and the aggregation can be performed more efficiently.

Considering the diameter of the network, with a FL algorithm, without packet size limitation, all data would have reached all nodes in 13 rounds. On this unstructured topology, TAS is clearly less efficient, since with the suboptimal wrap-up, about 65% of the data have been gathered, whereas with wrap-up using linear programming, between 60% of the data are contributing with a weight close to 1 and 90% with a non-zero weight.

B. TAS vs FL

In order to compare the TAS and the FL algorithms, we consider random trees and random unstructured topologies, with the same order of magnitude in terms of number of nodes.

For what concerns the analysis on random trees, we build a spanning tree on top of a random unstructured network, setting the inter-node communication range $d_{\mathrm{comm}} = \sqrt{\frac{\log_2 n}{2n}}$. According to [37], this range guarantees almost sure connectivity of a network of n nodes, deployed on a unit area. For each n (see the horizontal axis in Figure 6), 100 connected

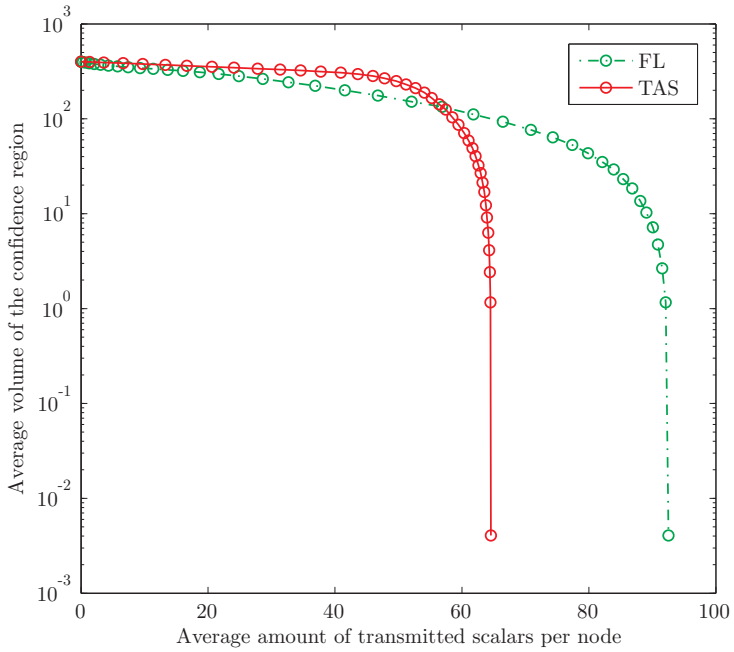

Fig. 7. Average volume, across nodes and 100 random tree realizations, of the 90% confidence region. Simulation parameters are set to $n = 100, n_{\mathrm{p}} = 2, q = 1$, and $m = 10$.

network realizations are instantiated. TAS and FL are compared in terms of the required number of data to be transmitted in each network realization. The success rate of TAS is the percentage of network realizations that proved favorable to TAS, *i.e.*, for which less measurements need to be exchanged to get all data reaching all nodes of the network.

Figure 6 shows this success rate as a function of n, for several values of n_{p}. As foreseen in the theoretical analysis in Section IV, there always exists a threshold value of n, depending on n_{p}, above which the TAS outperforms the FL algorithm, *i.e.*, the percentage closes to 100%.

We now investigate the trade-off between the confidence region volume and the amount of data transmitted by each node. Figure 7 shows the average volume of 90% confidence region as a function of the average amount of data that is communicated by each node. The volume and data amount are averaged across all nodes and across 100 random tree realizations, while simulation parameters are set to $n_{\mathrm{p}} = 2$, $q = 1$, $n = 100$ and $m = 10$. Figure 7 helps in determining the amount of data that needs to be transmitted by each node on average to obtain a given confidence region average volume. One can observe that the TAS algorithm outperforms the FL to achieve meaningful small volume values, in terms of the average amount of data transmitted by each node.

Finally, consider a random unstructured network, setting $n = 100$ and $n_{\mathrm{p}} = 3$. As shown

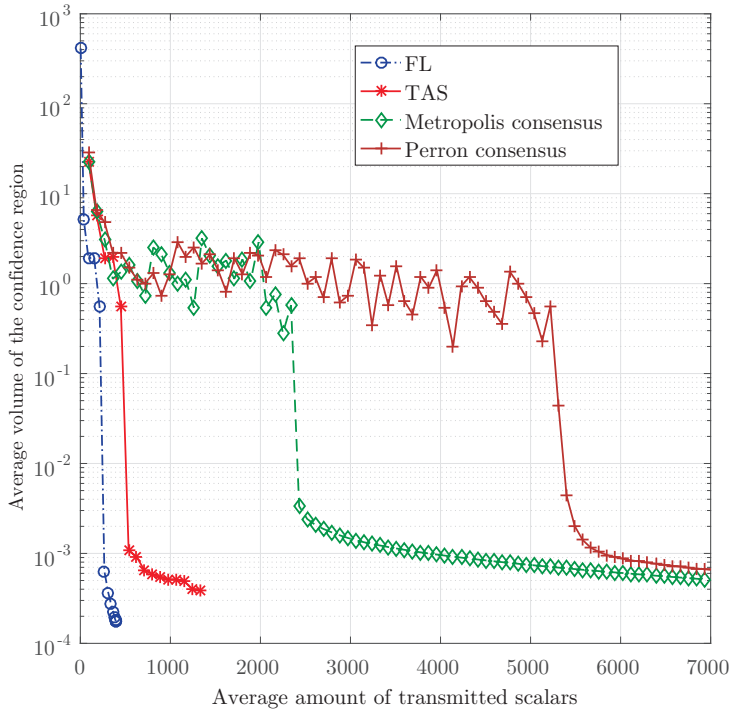

Fig. 8. Average volume, across nodes, of the 90% confidence region. A random unstructured network of 100 nodes is considered.

in Figure 8, the FL algorithm behaves better than TAS, providing lower volume values for the same amount of data. For comparison, it is also shown how both the FL and the TAS algorithm outperform the state-of-the-art consensus algorithms, independently of the considered consensus matrix (Metropolis [31] or Perron [11]).

This section confirms the general behavior that was highlighted in Section IV: On structured topologies, such as random trees there is an advantage in employing the TAS algorithm when the network dimension is sufficiently large. On unstructured networks of comparable size, the FL produces the best results, but, in any case, the absolute amount of data transmitted by each node is much larger than in structured networks. This suggests the adoption of structured networks, together with the TAS algorithm for the distributed computation of confidence regions, when the network traffic load for data diffusion is particularly critical.

VI. Experimental results

This section describes the practical implementation of both TAS and FL on the commercial sensor nodes EMB-Z2530PA [38] deployed in a real scenario. This implementation allows to account for the impact of the MAC layer.

A. Experimental setup

1) Network topologies: Two network topologies have been considered, namely (*i*) the flat network topology, where nodes can directly communicate by means of broadcast transmissions with their neighbours (that is, the nodes within their coverage region), and (*ii*) the random tree topology, in which a tree structure is randomly established by the nodes themselves at each run. For both topologies, the transmission power and the positions of nodes are managed in order to vary their connectivity level. In particular, for each network topology different measurement campaigns were carried out, varying the level of transmit power (the same for all nodes) in order to control the average (over the n nodes of the network) number n_{n} of neighbors of each node.

2) Network setup and data management: For both topologies, a network coordinator is introduced for monitoring and network setting purposes without compromising the distributed nature of the algorithms. At the beginning, the coordinator sends a *start* message that triggers the network setup (in the tree topology case), and the information diffusion algorithm, either FL or TAS.

For the tree topology, the tree construction starts from the root (level 0), which randomly selects the number n_{ch} of its children with uniform (discrete) distribution in $[1, 2, ..., n_{\mathrm{max}}]$. Provided that a sufficient number of nodes is available within the coverage range of the root, n_{ch} of them are selected as its children. Otherwise, all (thus less than n_{ch}) available nodes are joined to level 1. The same procedure is repeated by each node of level 1 and then iterated at all levels, until all nodes join the tree.

Once the network has been established, the information diffusion algorithm, either FL or TAS, is started, beginning from the leaves up to the root and then in the opposite direction. In our experimental setup the information transmitted by a node to its father is not overheard by its children.

In the flat network case, instead, no network-setup phase is needed. Hence either the FL or TAS execution is triggered as soon as the *start* packet is received.

For FL, each payload contains the amount of data transmitted, measurements and regressors, and a unique tag vector that identifies the contributing nodes. For TAS, payloads contain partial aggregated sums and a tag vector indicating the contributing nodes. In the proposed implementation, the tag consists of a vector of d_{TAG} bits, with 1s at the positions corresponding to the indexes of the contributing nodes. Since the same tag is used for TAS and FL algorithms, the difference in the transmission cost depends only on the amount of data transmitted.

3) Time axis management: At the beginning of each measurement period of duration T, the coordinator awakens all nodes, initiating the network activity. Whatever the network topology, the measurement period is divided into n_{r} rounds of equal duration $t_{\mathrm{r}} = \frac{T}{n_{\mathrm{r}}}$.

In the flat network case each node performs a measurement and, during each round, attempts to transmit.In the tree network case, instead, nodes are allowed to transmit only during the round pertaining to the level they belong to. Data (measurements and corresponding regressors for FL or aggregated sums for TAS) are then exchanged beginning from the leaves up to the root and then in the opposite direction.

To emulate the time jitter in nodes operations caused by local clocks drift in a distributed network as well as to avoid all nodes access the channel simultaneously, thus congesting the medium access control (MAC), each node defers the measurement phase, and therefore also the beginning of the information diffusion algorithm, by locally choosing a random delay $\Delta_i \in [0, t_{\mathrm{r}}]$, with $i = 1, 2, ..., n$.

All nodes stop data dissemination once n_{r} rounds have been completed. The coordinator collects then the amount of data transmitted/received by each node to allow an analysis of the behavior of the TAS and FL algorithms.

B. Results

A network of $n = 52$ nodes equipped with temperature sensors has been considered. The transmission power and the position of each node are chosen so that each node has an average number of neighbours n_{n} ranging from 2 to 33.

Simple temperature measurements are performed. The temperature θ^* is assumed constant in the area where the nodes are deployed. The corresponding measurement model is $y_i = \varphi_i\theta^* + w_i$, where φ_i is known by each node,[5] and θ^* is the parameter to estimate. Thus $n_{\mathrm{p}} = 1$ and the data to be transmitted by the FL algorithm are collections of pairs $(\boldsymbol{\varphi}_i, y_i)$, consisting in this case of $d_{\mathrm{FL}} = 2$ real values (which may be quantized). For the SPS algorithm, one chooses $m = 10$, and $q = 1$ to be able to characterize 90% confidence regions according to (8). Therefore, the amount of data transmitted at each round by TAS is $d_{\mathrm{TAS}} = 20$ real values (which may also be quantized) and remains constant.

[5]Here, for simplicity, we choose $\varphi_i = 1\ \forall i$. However, this choice does not affect the outcomes of our investigations. With a larger number of sensors it would be possible to estimate also spatial variations of the temperature, but the simple example here considered is enough for the purpose of this paper.

Parameter	Symbol	value
Number of nodes	n	52
Maximun number of children (tree topology only)	$n_{\max}$	5
Measurement period	T	2 s
Number of rounds	n_{r}	$\{2, 3, \ldots, 30\}$
Number of neighbours	n_{n}	$\{2, 4, 8, 17, 33\}$
Number of parameters to be estimated	n_{p}	1
Number of sign perturbed sums	m	10
Size of data sets with FL	d_{F}	2 Bytes
Size of data sets with TAS	d_{TAS}	20 Bytes
Size of the tag vector (both TAS and FL)	d_{TAG}	7 Bytes

TABLE V

PARAMETERS OF THE EXPERIMENTAL SETUP

The measurement period is taken as $T = 2$ s. n_{r} ranges from 2 to 30 and therefore t_{r} varies from 1 s down to 67 ms. The parameters adopted for our experimental campaign are summarized in Table V.

Given the network topology (either generic tree or flat network) and for each chosen setup (transmit power, n_{r}), we performed the measurement campaign over 100 network realizations and we derived the average (over the 100 resulting networks) amount of information received by each node and the average amount of data transmitted in the whole network.

1) Flat network: Figure 9 shows the average proportion (expressed in percentage) of data reaching a given node in a flat topology for various n_{r} and n_{n}.

The value of n_{r} that maximizes the average amount of received data depends on n_{n}. For low values of n_{r}, the performance is limited by the constraint on the maximum number of allowed hops (that is coincident with n_{r}), that might not be sufficient for a particular data to reach all nodes in the network, especially for low degrees of connectivity n_{n}. On the contrary, for large values of n_{r} the performance is limited by the MAC, as a small t_{r} increases the collision probability.

From the same figure one can also see that better performances are obtained when the network is characterized by a low degree of connectivity n_{n} provided that a sufficiently high number of rounds can be allocated within the measurement period. In fact, large n_{n}, *i.e.*, high power levels, generate more interference among nodes that leads the MAC to collapse. This

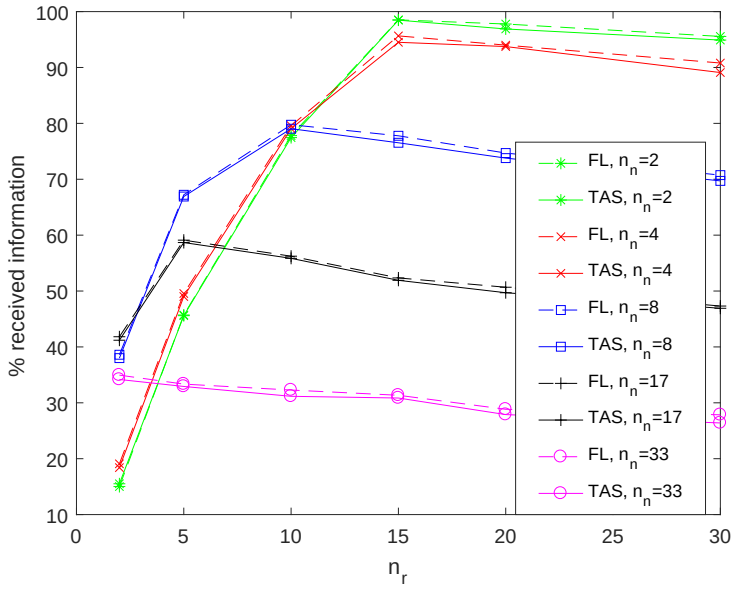

Fig. 9. Flat network: average proportion of the total information received by nodes as a function of n_{r} for various n_{n}. The legend entries and the curves in the right-hand part of the figure are in the same order.

Neighbors	FL Experimental	TAS Experimental
2	5236	14337
4	5171	12770
8	4197	8770
17	2832	4860
33	1705	2561

TABLE VI

FLAT NETWORK: AVERAGE AMOUNT OF TRANSMITTED DATA (SCALARS) WITHIN THE WHOLE NETWORK IN THE CASE $n_{\mathrm{R}} = 15$.

suggests that a proper power control strategy able to keep n_{n} at minimum values to keep connectivity is beneficial both for network performance as well as to save energy.

FL and TAS perform similarly in all conditions, hence they are equivalent considering only the amount of received information. They differ, instead, in terms of amount of transmitted information, as seen in Table VI, which reports the average amount (over 100 network realizations) of transmitted data (scalars) within the whole network in the case $n_{\mathrm{r}} = 15$.

When operated in a flat topology, FL outperforms TAS as it requires a lower amount of transmitted information. With such topology, in fact, the information efficiently diffuses within the network, up to the maximum extent permitted by the transmission power and without back and forth paths (that occur, instead, in the tree topology), hence the aggregation

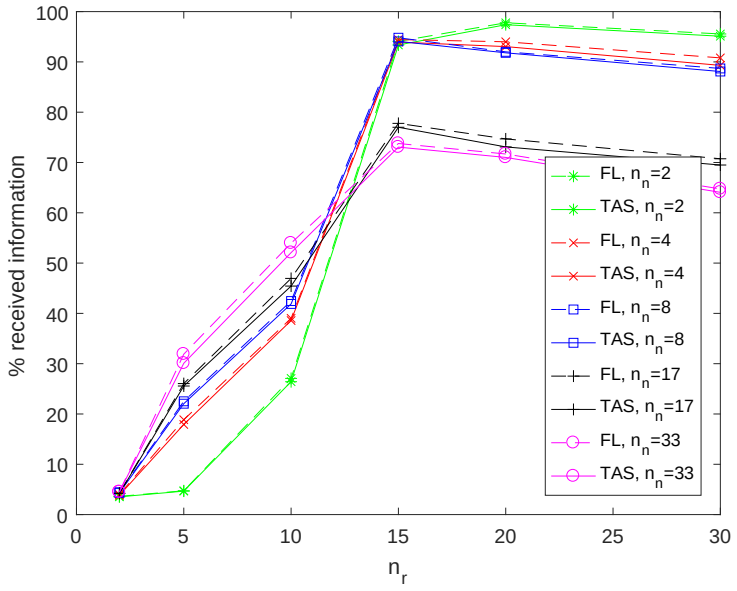

Fig. 10. Generic tree topology: Average percentage of information received by a node. The legend entries and the curves in the right-hand part of the figure are in the same order.

carried out by TAS is not sufficient to compensate the larger value of d_{TAS} with respect to d_{FL}.

2) Generic tree topology: Figure 10 provides for the tree topology the average proportion (expressed in percentage) of data reaching a given node as a function of n_{r} for various n_{n}. Here it can be noticed a limited sensitivity of the optimum value of n_{r} to n_{n}, as the average number of children of each node only slightly depends on the connectivity degree. In fact, for the tree topology in this example we upper bounded by $n_{\text{max}} = 5$ the number of children of each node to avoid the generation of 'fat' trees. Therefore, for a given node only a fraction of its neighbors are actually involved in data diffusion. As a consequence, increasing the number of neighbors n_{n} does not increase the amount of information diffused, but determines higher levels of interference and packet collisions.

This makes power control less critical in tree topologies with respect to flat topologies. In general, better data dissemination is observed when n_{r} is large compared to flat topologies since transmissions happen level by level and only a small part of the network tries to access the channel at the same time. On the contrary, with small values of n_{r}, data disseminate only to a limited part of the network due to the depth of the tree which may be larger[6] than $1 + n_{\text{r}}/2$. Similarly to the flat topology, even in this case FL and TAS are very similar

[6]With n_{r} rounds the maximum number of levels of a tree that allows a complete dissemination of data from the leaves up to the root and back is $1 + n_{\text{r}}/2$.

Neighbors	FL analytic	FL exp.	TAS analytic	TAS exp.
2	2179	2047	1427	1330
4	2144	2022	1420	1331
8	2087	1978	1409	1322
17	1802	1400	1353	1042
33	1705	1256	1334	972

TABLE VII

GENERIC TREE TOPOLOGY: AVERAGE PROPORTION OF TRANSMITTED DATA REACHING A GIVEN NODE AS A FUNCTION OF n_{R} FOR VARIOUS n_{N}.

in terms of amount of received data. Table VII reports the average amount of transmitted data within the whole network when $n_{\mathrm{r}} = 15$. Now, TAS outperforms FL when operated on a tree topology. Table VII also compares the analytical outcomes, derived feeding (25) and (26) with the parameters corresponding to each network realization and averaging over all realizations, and the respective averages of experimental results. When the number of neighbors is small ($n_{\mathrm{n}} = 2, 4, 8$) a good agreement between analysis and measurements is observed both for TAS and FL. The experimental values are always less than the analytical ones because, as can be observed in Figure 10, the amount of received information never reaches 100%, even in the considered case of $n_{\mathrm{r}} = 15$.

This phenomenon is emphasized as n_{n} increases ($n_{\mathrm{n}} = 17, 33$), which further reduces the amount of received data (Figure 10) and hence the amount of data transmitted by nodes with respect to the ideal (no collisions) situation described by the analysis.

We can conclude, therefore, that the analytical framework can be usefully exploited to provide performance predictions in not congested networks and a performance bound in MAC limited networks.

To evaluate the influence of the proportion of measurements received by each node on the quality of the confidence region that can be derived, a temperature measurement has been performed by each of the n nodes of the network. For different target proportions $\rho \in [0, 1]$ of measurements reaching some node of the network, 100 random selections of a subset of measurements have been considered and a $90\ \%$ confidence region evaluation with $m = 10$ and $q = 1$ has been performed. Figure 11 describes the evolution of the average width of the 90 % confidence region as a function of the proportion of measurements collected by a given node. Figure 11 (right) shows that the width decreases approximately as $1/\sqrt{\rho n}$, which is consistent with what is observed when maximum-likelihood estimation is carried

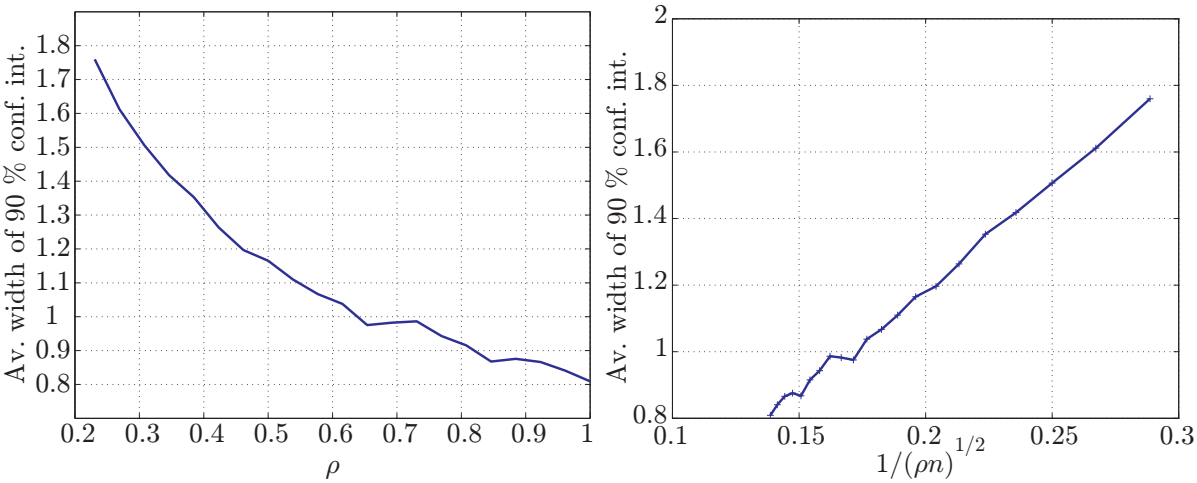

Fig. 11. Average width of the 90 % confidence region as a function of the proportion ρ (left) of measurements collected by a node with FL or TAS and as a function $1/\sqrt{\rho n}$ (right).

out assuming an additive Gaussian noise [8], although this hypothesis on the noise is not considered here. From Figures 9 or 10 and 11, one may deduce the width of the confidence interval that may be obtained with FL or TAS, when not all measurements have reached some node. One can for example see that even if only 80 % of the measurements have reached a node, the width of the confidence region is only 10 % larger than that obtained from all measurements. This means that if one tolerates evaluating a confidence region from a reduced subset of the data, the constraints on the data dissemination duration may be significantly relaxed, with beneficial effects in terms of time and energy savings.

VII. Conclusions

This paper investigates the distributed evaluation of non-asymptotic confidence regions at each node in a sensor network. It presents the TAS algorithm and its comparison with other information diffusion algorithms on structured and unstructured topologies. The TAS algorithm has been designed to efficiently exploit the peculiarities of the distributed evaluation of confidence regions via SPS. Simulation results provide a characterization of the trade-off for the achievable average confidence region volume as a function of the required amount of data that each node should transmit on average. The contributions show that, on structured networks, the proposed TAS algorithm is able to outperform the FL when the network dimension is sufficiently high, this independently of the specific dimension of the parameter space, as investigated in the theoretical and numerical analyzes, as well as on an experimental setup.

APPENDIX A
TAS PSEUDO CODE

The pseudo-codes for each phase of the TAS algorithm are reported in Algorithms 2 to 7. The TAS algorithm is run similarly at each node of the network. The superscript $^{(k)}$ is thus omitted to lighten notations. All variables are assumed to be global.

Algorithm 2 Initialization

▷Get local sensor measurement◁

$y_k \leftarrow$ `PerformMeasurement`

▷Format data and transmit to neighbors◁

create tag vector $\mathbf{t}$ according to (16)

create data vector $\boldsymbol{\delta}$ according to (17)

`TransmitToNeighbors` $(\mathbf{t}, \boldsymbol{\delta})$

▷Initialize $\mathbf{R}$ with local infos◁

$\mathbf{R}.\mathbf{T} = \mathbf{t}$

$\mathbf{R}.\mathbf{D} = \boldsymbol{\delta}$

Algorithm 3 Reception

▷Get node indexes from which packets are received◁

idx $\leftarrow$ `GetNodeIdx`

▷Update reception structure $\mathbf{Rx}$ with the tags and partial sums received from neighbors stored in $Node(i).\mathbf{t}$ and $Node(i).\boldsymbol{\delta}$◁

for `i=1 to length(`**idx**`)` **do**

    $\mathbf{Rx}.\mathbf{T} \leftarrow [\mathbf{Rx}.\mathbf{T}; Node(\mathbf{idx}(i)).\mathbf{t}]$

    $\mathbf{Rx}.\mathbf{D} \leftarrow [\mathbf{Rx}.\mathbf{D}; Node(\mathbf{idx}(i)).\boldsymbol{\delta}]$

end for

Algorithm 4 Distillation

▷Distillation of new and already stored infos◁

1: **for** lx=1 to NbRows($\mathbf{Rx}.\mathbf{T}$) **do**
2: **for** l=1 to NbRows($\mathbf{R}.\mathbf{T}$) **do**
3: **if** $\mathbf{R}.\mathbf{T}(\mathtt{l}) \subset \mathbf{Rx}.\mathbf{T}(\mathtt{lx})$ **then**

▷Clear received packet from already stored data◁

4: $\mathbf{Rx}.\mathbf{T}(\mathtt{lx}) \leftarrow \mathbf{Rx}.\mathbf{T}(\mathtt{lx}) - \mathbf{R}.\mathbf{T}(\mathtt{l})$
5: $\mathbf{Rx}.\mathbf{D}(\mathtt{lx}) \leftarrow \mathbf{Rx}.\mathbf{D}(\mathtt{lx}) - \mathbf{R}.\mathbf{D}(\mathtt{l})$
6: **end if**
7: **if** $\mathbf{Rx}.\mathbf{T}(\mathtt{lx}) \subset \mathbf{R}.\mathbf{T}(\mathtt{l})$ **then**

▷Clear already stored data from received data◁

8: $\mathbf{R}.\mathbf{T}(\mathtt{l}) \leftarrow \mathbf{R}.\mathbf{T}(\mathtt{l}) - \mathbf{Rx}.\mathbf{T}(\mathtt{lx})$
9: $\mathbf{R}.\mathbf{D}(\mathtt{l}) \leftarrow \mathbf{R}.\mathbf{D}(\mathtt{l}) - \mathbf{Rx}.\mathbf{D}(\mathtt{lx})$
10: **end if**
11: **end for**

▷Any distilled received data is appended to $\mathbf{R}$ ◁

12: **if** $\mathbf{Rx}.\mathbf{T}(lx) \neq \mathbf{0}$ **then**
13: $\mathbf{R}.\mathbf{T} \leftarrow [\mathbf{R}.\mathbf{T}; \mathbf{Rx}.\mathbf{T}(\mathtt{lx})]$
14: $\mathbf{R}.\mathbf{D} \leftarrow [\mathbf{R}.\mathbf{D}; \mathbf{Rx}.\mathbf{D}(\mathtt{lx})]$
15: **end if**
16: **end for**

▷Clear reception structure of current node◁

17: **clear** $\mathbf{Rx}$

Algorithm 5 Aggregation

▷ Perform aggregation of Tags and partial sums. Using boolean flag vector $\mathbf{Agd}$, already aggregated infos are no more considered for aggregation in subsequent rounds ◁

$\mathbf{t} \leftarrow \mathbf{0}$ ▷ Initialize aggregated tag vector
$\boldsymbol{\delta} \leftarrow \mathbf{0}$ ▷ Initialize aggregated data vector
for `l=1 to NbRows`$(\mathbf{R}.\mathbf{T})$ **do**
  if $\mathbf{Agd}(\mathtt{l}) = \textit{false}$ **then**
    if $\mathbf{R}.\mathbf{T}(\mathtt{l}) \cap \mathbf{t} = \mathbf{0}$ **then**
      $\mathbf{t} \leftarrow \mathbf{t} + \mathbf{R}.\mathbf{T}(\mathtt{l})$
      $\boldsymbol{\delta} \leftarrow \boldsymbol{\delta} + \mathbf{R}.\mathbf{D}(\mathtt{l})$
      $\mathbf{Agd}(\mathtt{l}) = \textit{true}$ ▷ $\mathtt{l}$-th row of $\mathbf{R}.\mathbf{T}$ flagged as aggregated
    end if
  end if
end for

Algorithm 6 Transmission

if $\mathbf{t} \neq \mathbf{0}$ **then**
  `TransmitToNeighbors` $(\mathbf{t}, \boldsymbol{\delta})$
end if

Algorithm 7 Wrap-up

▷Sorts lines of $\mathbf{R}$ by decreasing weight of lines of $\mathbf{R}.\mathbf{T}$◁

▷ Perform aggregation of tags and partial sums.◁

$\mathbf{t} \leftarrow \mathbf{0}$ ▷ Initialize wrapped-up tag vector
$\boldsymbol{\delta} \leftarrow \mathbf{0}$ ▷ Initialize wrapped-up data vector
for `l=1 to NbRows`$(\mathbf{R}.\mathbf{T})$ **do**
  if $\mathbf{R}.\mathbf{T}(\mathtt{l}) \cap \mathbf{t} = \mathbf{0}$ **then**
    $\mathbf{t} \leftarrow \mathbf{t} + \mathbf{R}.\mathbf{T}(\mathtt{l})$
    $\boldsymbol{\delta} \leftarrow \boldsymbol{\delta} + \mathbf{R}.\mathbf{D}(\mathtt{l})$
  end if
end for

REFERENCES

[1] V. Zambianchi, M. Kieffer, F. Bassi, G. Pasolini, and D. Dardari, "Distributed SPS algorithms for non-asymptotic confidence region evaluation," in *Proc. of the 23rd European Conference on Networking and Communication, EUCNC 2014*, Bologna, Italy, June 2014, pp. 1–5.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, March 2002.

[3] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proc. of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, New York, USA, September 2002, pp. 88–97.

[4] R. Verdone, D. Dardari, G. Mazzini, and A. Conti, in *Wireless Sensor and Actuator Networks: technologies, analysis and design*. Elsevier Ltd, London, 2008.

[5] T. Quek, D. Dardari, and M. Win, "Energy efficiency of dense wireless sensor networks: to cooperate or not to cooperate," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 2, pp. 459–470, February 2007.

[6] J. Matamoros, F. Fabbri, C. Anton-Haro, and D. Dardari, "On the estimation of randomly sampled 2D spatial fields under bandwidth constraints," *IEEE Trans. Wireless Commun.*, vol. 10, no. 12, pp. 4184–4192, Dec. 2011.

[7] G. Mao, B. Fidan, and B. D. Anderson, "Wireless sensor network localization techniques," *Computer Networks*, vol. 51, no. 10, pp. 2529 – 2553, 2007.

[8] S. M. Kay, *Fundamentals of Statistical Processing: Estimation Theory*. Prentice-Hall Signal Processing Series, 1993, vol. 1.

[9] I. D. Schizas, G. Mateos, and G. B. Giannakis, "Distributed LMS for consensus-based in-network adaptive processing," *IEEE Trans. Signal Process.*, vol. 57, no. 6, pp. 2365–2382, June 2009.

[10] G. Mateos, I. D. Schizas, and G. B. Giannakis, "Distributed recursive least-squares for consensus-based in-network adaptive estimation," *IEEE Trans. Signal Process.*, vol. 57, no. 11, pp. 4583–4588, November 2009.

[11] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, January 2007.

[12] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks," in *Proc. of the 46th IEEE Conference on Decision and Control*, New Orleans, USA, December 2007, pp. 5492–5498.

[13] ——, "Kalman-consensus filter: Optimality, stability, and performance," in *Proc. of the 48th IEEE Conference on Decision and Control*, Shanghai, China, December 2009, pp. 7036–7042.

[14] B. Yang and J. Scheuing, "Cramér-Rao bound and optimum sensor array for source localization from time differences of arrival," in *Proc. of IEEE Acoustics, Speech, and Signal Processing, ICASSP*, Philadelphia, PA, USA, March 2005, pp. 961–964.

[15] X. Sheng and Y.-H. Hu, "Maximum likelihood multiple-source localization using acoustic energy measurements with wireless sensor networks," *IEEE Trans. Signal Process.*, vol. 53, no. 1, pp. 44–53, Jan 2005.

[16] N. Patwari, J. Ash, S. Kyperountas, A. Hero, R. Moses, and N. Correal, "Locating the nodes: cooperative localization in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 22, no. 4, pp. 54–69, July 2005.

[17] D. Jourdan, D. Dardari, and M. Win, "Position error bound for UWB localization in dense cluttered environments," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 44, no. 2, pp. 613–628, April 2008.

[18] M. C. Campi and E. Weyer, "Guaranteed non-asymptotic confidence regions in system identification," *Automatica*, vol. 41, no. 10, pp. 1751–1764, October 2005.

[19] M. C. Campi, S. Ko, and E. Weyer, "Non-asymptotic confidence regions for model parameters in the presence of unmodelled dynamics," *Automatica*, vol. 45, no. 10, pp. 2175–2186, October 2009.

[20] B. C. Csáji, M. C. Campi, and E. Weyer, "Non-asymptotic confidence regions for the least-squares estimate," in *Proc. of IFAC Symposium on System Identification, SYSID*, Brussels, Belgium, July 2012, pp. 227–232.

[21] ——, "Signed-Perturbed Sums (SPS): a method for constructing exact finite-sample confidence regions for general linear systems," in *Proc. of the 51th IEEE Conference on Decision and Control*, Maui, Hawaii, USA, December 2012, pp. 7321–7326.

[22] ——, "Signed-Perturbed Sums: A new system identification approach for constructing exact non-asymptotic confidence regions in linearregression models," *IEEE Trans. on Signal Processing*, vol. 63, no. 1, pp. 169–181, January 2015.

[23] S. Kolumban, I. Vajk, and J. Schoukens, "Perturbed datasets methods for hypothesis testing and structure of corresponding confidence sets," *Automatica*, vol. 51, pp. 326–331, January 2015.

[24] M. Kieffer and E. Walter, "Guaranteed characterization of exact non-asymptotic confidence regions as defined by LSCR and SPS," *Automatica*, vol. 50, no. 2, pp. 507–512, February 2014.

[25] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *Proc. of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, ser. MobiCom '99. Seattle, WA, USA: ACM, August 1999, pp. 174–185.

[26] A. Nordio, C. Chiasserini, and E. Viterbo, "Performance of linear field reconstruction techniques with noise and uncertain sensor locations," *IEEE Trans. Signal Process.*, vol. 56, no. 8, pp. 3535–3547, Aug 2008.

[27] C. Perkins and E. Royer. (2003) Ad hoc on-demand distance vector (aodv) routing. RFC 3561.

[28] E. Walter and L. Pronzato, *Identification of Parametric Models from Experimental Data*. London: Springer-Verlag, 1997.

[29] S.-Y. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inf. Theory*, vol. 49, no. 2, pp. 371–381, Feb 2003.

[30] R. Koetter and M. Medard, "An algebraic approach to network coding," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 782–795, Oct 2003.

[31] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Syst. Control Lett.*, vol. 53, no. 1, pp. 65–78, September 2004.

[32] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proc. of the Fourth International Symposium on Information Processing in Sensor Networks, IPSN 2005*, Los Angeles, USA, April 2005, pp. 63–70.

[33] J.-J. Xiao, A. Ribeiro, Z.-Q. Luo, and G. Giannakis, "Distributed compression-estimation using wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 23, no. 4, pp. 27–41, 2006.

[34] L. Xiao, S. Boyd, and S. J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of Parallel and Distributing Computing*, vol. 67, no. 1, pp. 34–46, 2007.

[35] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "An adaptive energy-efficient and low-latency MAC for tree-based data gathering in sensor networks," *Wireless Communications and Mobile Computing*, vol. 7, no. 7, pp. 863–875, 2007. [Online]. Available: http://dx.doi.org/10.1002/wcm.503

[36] S. Rump, "INTLAB - INTerval LABoratory," in *Developments in Reliable Computing*, T. Csendes, Ed. Dordrecht: Kluwer Academic Publishers, 1999, pp. 77–104. [Online]. Available: http://www.ti3.tu-harburg.de/rump/

[37] P. Gupta and P. Kumar, "The capacity of wireless networks," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 388–404, Mar 2000.

[38] EMBIT. (2016, Apr.) EMB-Z2530PA. [Online]. Available: http://www.embit.eu/products/wireless-modules/emb-z2530pa/