Distributed Fault Detection in Sensor Networks via Clustering and Consensus

Gianluca Bianchin, Angelo Cenedese, Michele Luvisotto, and Giulia Michieletto

Abstract— **In this paper we address the average consensus problem in a Wireless Sensor-Actor Network with the particular focus on autonomous fault detection. To this aim, we design a distributed clustering procedure that partitions the network into clusters according to both similarity of measurements and communication connectivity. The exploitation of clustering techniques in consensus computation allows to obtain the detection and isolation of faulty nodes, thus assuring the convergence of the other nodes to the exact consensus value. More interestingly, the algorithm can be integrated into a Kalman filtering framework to perform distributed estimation of a dynamic quantity in presence of faults. The proposed approach is validated through numerical simulations and tests on a real world scenario dataset.**

I. Introduction

Wireless Sensor-Actor Networks (WSANs) are systems composed of a large number of interactive devices distributed over a vast area, linked by a wireless medium and equipped with computational capabilities [1]. In recent years, the definition of algorithms and models to manage such networked systems has emerged among the most popular and prolific research topics. At the same time, the application domains of large scale sensor networks are growing and range from factory automation to ambient assisted living [2], [3], [4]: while the characteristics of the devices employed in the specific applications may be quite different, WSANs are devised according to common criteria and employ collaborative strategies, in order to ensure remarkable features such as scalability and robustness. To model and design such properties, a distributed approach is often preferred to a centralized one, and many strategies have been lately studied and developed that rely on network nodes cooperation to perform global tasks resting upon local rules.

In this context, a well-known example of a distributed algorithm is the *linear average consensus*, which guarantees the convergence of node states (e.g. sensor measurements of a quantity of interest) to the average of their initial values through information exchange at local level [5], [6], [7], [8]. Interestingly, clustering may result to be an appealing approach to be employed with consensus since grouping nodes into clusters can improve the convergence properties by better exploiting the network connectivity and the similarities among nodes. To this aim, many efficient strategies have been proposed to partition a WSAN according to a specific definition of clusters, as for example in [9] and [10].

From the application point of view, clustering may result interesting also for fault detection and faulty node isolation: indeed, if not recognized, a faulty sensor (be it for a malicious behavior or for an actual failure) can compromise the operation of the whole network and prevent the attainment of the desired task. This problem has been deeply studied and several algorithms based on clustering approach are available in literature [11], [12], [13]. Nonetheless, to the best of our knowledge, clustering techniques have not yet been applied to distributed fault detection for secure consensus computation although the scientific community has shown a strong interest towards this topic [14], [15], [16].

In this work, the distributed fault detection problem in the average consensus framework is addressed and, to solve this issue, a strategy is proposed that exploits a clustering procedure based on both similarity of measurements and communication connectivity. The effectiveness of the approach is supported by numerical results obtained from highly connected sensor networks. A validation on data gathered from an actual WSAN installation in a real-world environment has also been conducted.

The remainder of the paper is organized as follows. After some preliminary review of useful network notions in Sec. II, the distributed clustering procedure is presented in Sec. III and it is applied to solve the average consensus problem in presence of faulty nodes. Then, Sec. IV is dedicated to the validation of the cluster-based consensus algorithm through its application to the interesting case of a measurement flow related to a dynamic process. Similarly, Sec. V deals with tests on data from a real scenario. Finally, in Sec. VI some concluding remarks are drawn with respect to current and future developments.

II. Network Notions and Definitions

Let a WSAN be modeled by a network of N nodes endowed with a set of states $\mathbf{x} = [x_1 \dots x_N]^\top$, that are scalar noisy measurements of a global quantity $x \in \mathbb{R}$ that the N agents attempt to estimate. The network topology is represented by a highly connected undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ are the sets of nodes and edges, respectively. We define the *adjacency matrix* E such that $E(i,j) = 1$ if and only if $(i,j) \in \mathcal{E}$, the *neighborhood* of a node i as $\mathcal{N}(i) = \{j \in \mathcal{V} : E(i,j) = 1, i \neq j\}$ and its *degree* as $d(i) = |\mathcal{N}(i)|$.

A.Cenedese, M.Luvisotto, and G.Michieletto are with the Department of Information Engineering, University of Padova, via Gradenigo 6/B, 35131 Padova, Italy. G.Bianchin is with the Department of Mechanical Engineering, University of California Riverside, CA (USA). Corresponding author: A.Cenedese, angelo.cenedese@unipd.it.

The research leading to these results was partially supported by MIUR Project SEAL - Smart&safe Energy-aware Assisted Living (SCN-00398).

This technical report is an extended version of the work presented at the Conference on Decision and Control 2015 (CDC15). In case of citation, please cite the CDC paper (same authors and title).

The average of the nodes measurements, $\hat{x} = \frac{1}{N}\sum_i x_i$, is usually considered to be a reliable estimation for the quantity of interest and can be computed in a distributed way through the *average consensus algorithm*. Following this strategy, agents iteratively update their estimates according to

$$\mathbf{x}(k+1) = P\mathbf{x}(k) \tag{1}$$

where $\mathbf{x}(k) = [x_1(k) \dots x_N(k)]^\top$ is the vector of nodes estimates at k-th iteration, $\mathbf{x}(0) = \mathbf{x}$ is the vector of initial measurements and $P \in \mathbb{R}^{N\times N}$ is a weight matrix. If P is doubly stochastic, this procedure is ensured to converge, i.e. $\lim_{k\to\infty} x_i(k) = \hat{x}\ \forall i$. Among the several strategies that are available to build a matrix that satisfies this condition, in this work we adopt the Metropolis-Hastings weights [17], i.e.

$$P(i,j) = \begin{cases} \frac{1}{max(d(i),d(j))+1} & \text{if } i \neq j \text{ and } E(i,j) = 1 \\ 1 - \sum_{j=1, i\neq j}^{N} P(i,j) & \text{if } i = j \text{ and } E(i,j) = 1 \\ 0 & \text{if } E(i,j) = 0 \end{cases} \tag{2}$$

which show good convergence properties and are particularly suitable for a distributed implementation. Indeed, in such a case, the i-th node that computes row $P(i,:)$ needs to know only its own degree $d(i)$ and those of its neighbors.

Given these premises, we introduce the following

***Definition** 1 (Cluster):* A *cluster* in the network $\mathcal{G}$ is a group $\mathcal{C} = \{c_1, \dots, c_k\} \subseteq \mathcal{V}$ of $k \leq N$ nodes such that the following criteria are both satisfied.

C1 *Cluster connectivity*: for each pair of nodes c_i, c_j in $\mathcal{C}$, there exists at least one path connecting them entirely composed by elements of the same cluster;

C2 *Measurement similarity*: for each pair of nodes c_i, c_j in $\mathcal{C}$, it must hold $|x_{c_i} - x_{c_j}| < b$, where b is the imposed *clustering bound* (setup parameter). □

In other words, if two nodes have similar measurements but are not connected either directly or through nodes of the same cluster, they are not clustered together: the communication between them would not be managed by a single cluster and hence it is not judged reliable. On the other hand, if two nodes are connected but show dissimilar measurements, it is assumed that they are not observing the same phenomenon and so they are assigned to different clusters.

In practical implementations, the consensus algorithm (1) is strictly dependent on the measurements and communication capabilities and, consequently, the presence of faulty nodes can cause system misbehaviors, performance losses or even compromise the successful completion of a task. As a general definition, a fault occurs *when the delivered service deviates from the expected service* [18]. Hereafter, a specific fault model is considered, related to a wrong measurement in the WSAN context.

***Definition** 2 (Fault):* Let $x \in \mathbb{R}$ be the true value of a quantity to measure, a sensor i is considered *faulty* if its noisy measurement $x_i = x + w_i$ (w_i being some noise realization) does not fall in a range $\mathcal{R} = [x - \delta, x + \delta]$, where δ is a chosen design parameter[1]. □

According to this definition, a sensor that permanently or temporarily provides an unreliable measurement is faulty and it always influences the estimation of other nodes in a standard consensus procedure (1), forcing the global estimation to be biased with respect to the true value.

III. Distributed Clustering Algorithm for Fault Detection

In this framework, a procedure has been developed that partitions the network into a (non-fixed) number of clusters that fulfill criteria C1-C2, and that can be then exploited to handle the fault detection and isolation problem in a WSAN.

A. Algorithm Outline

The proposed algorithm is based on the circulation of a special *token* message over the network, which starts from a randomly selected root node and enables the node that receives it to temporarily exchange information with its neighbors and update its internal information.

[1]In practice, suitable values for δ can be derived from the sensitivity of employed sensors, reported in technical specifications, or from ad hoc experimental tuning.

Algorithm 1

```
1:  procedure DISTRIBUTED CLUSTERING
2:      $term \leftarrow false$
3:      while not $term$ do
4:          if not $v(i)$ then
5:              $v(i) \leftarrow true$
6:              if not $k(i)$ then
7:                  $k(i) \leftarrow true$
8:              end if
9:              for $j \leftarrow 1$ to $N$, $j \neq i$ do
10:                 if $E(i,j) = 1$ and $j \notin \mathcal{C}(i)$ then
11:                     if not $k(j)$ then
12:                         $\mathcal{Q}(i) \leftarrow enqueue(j)$
13:                     end if
14:                     if $|x_k - x_j| < b\ \forall k \in \mathcal{C}(i)$ then
15:                         $\mathcal{C}(i) \leftarrow \mathcal{C}(i) \cup j$
16:                         $k(j) \leftarrow true$
17:                     end if
18:                 end if
19:             end for
20:         end if
21:         if $\mathcal{Q}(i) \neq \emptyset$ then
22:             $next \leftarrow dequeue(\mathcal{Q}(i))$
23:             $r(next) \leftarrow i$
24:         else
25:             if $r(i) \neq 0$ then
26:                 $next \leftarrow r(i)$
27:             else
28:                 $term \leftarrow true$
29:             end if
30:         end if
31:         if $next \in \mathcal{C}(i)$ and not $term$ then
32:             update $\mathcal{C}(next)$ with elements of $\mathcal{C}(i)$
33:         end if
34:         $i \leftarrow next$
35:     end while
36: end procedure
```

This procedure is reported as Alg. 1, where the following variables are defined for each node i:
- $\mathcal{Q}(i)$, queue containing next nodes to transmit the token,
- $k(i)$, flag indicating if node has been clustered,
- $\mathcal{C}(i)$, indexes of nodes belonging to the same cluster as i,
- $r(i)$, node whence the token has been received,
- $v(i)$, flag indicating if node has received the token for the first time.

The initialization requires setting $\mathcal{Q}(i) = \emptyset$, $\mathcal{C}(i) = \{i\}$, $r(i) = 0$, $v(i) = false$ and $k(i) = false$.

The first time the i-th node receives the token (rows 4-20, *part A*), it retrieves the measurements of its neighbors and if the *measurement similarity* condition (C2) is satisfied, it adds them to $\mathcal{C}(i)$. Then, every time node i has the token (rows 21-34, *part B*), it selects the next node j to pass the token, either dequeuing $\mathcal{Q}(i)$ or, if $\mathcal{Q}(i) = \emptyset$, setting $j = r(i)$. If $j \in \mathcal{C}(i)$, the cluster of the next node is updated with the elements of $\mathcal{C}(i)$. The procedure terminates when all nodes have been visited.

B. Algorithm Analysis

Focusing on the algorithm structure, it is possible to compute its complexity in terms of communication exchanges between nodes and to prove its correctness.

***Theorem** 1 (Complexity):* Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a network with cardinality $|\mathcal{V}| = N$. The distributed clustering algorithm Alg. 1 has complexity of $\mathcal{O}(N^2)$ in terms of communication exchanges.

Proof: The first part (*part A*) of the procedure is executed once for each node i. During its execution, for each neighbor j, there are at least two and at most three exchanges of information. Indeed, node i requires information from node j, which sends its measurement x_j and the value of $k(j)$. If the similarity condition is satisfied, node i sends to node j the updated value of $k(j)$. The edge between nodes i and j, therefore, is explored at most three times when i has the token and three more times when j has it. In the worst case, each edge is explored six times, so the complexity is $\mathcal{O}(|\mathcal{E}|)$. By reminding that the number of edges in a complete graph is given by $\frac{N(N-1)}{2}$, if we approximate $|\mathcal{E}|$ with $\frac{N^2}{2}$, the total number of communication exchanges in this part of the algorithm is bounded by $3N^2$.

The second part (*part B*) of the algorithm requires only one communication exchange. Each node i can receive the token from node $r(i)$ (case 1) or from a node in its queue (case 2). Let us consider the worst case, i.e. each node is in a different cluster and the graph is complete. In this case, node 1 inserts nodes $2, \dots, N$ in its queue, node 2 inserts nodes $3, \dots, N$ and so on. Each node i encounters $i - 1$ times case 1 and $N - i$ times case 2, so in total the second part is executed $N - 1$ times for each node. The total number of communication exchanges is $N(N-1) \leq N^2$, hence the complexity of this part is again $\mathcal{O}(N^2)$.

The overall complexity of the algorithm is therefore $\mathcal{O}(N^2)$ and the upper bound for the total number of communications is given by $4N^2$. ■

***Theorem** 2 (Correctness):* The distributed clustering procedure Alg. 1 partitions a N-nodes network into a certain number of clusters according to *cluster connectivity* and *measurement similarity* from Def. 1.

Proof: To prove the algorithm correctness it is necessary to show that, if a generic node j belongs to $\mathcal{C}(i)$, the following two conditions must hold:

(a) *Cluster connectivity*: there exists a path linking node i and node j composed only by nodes in $\mathcal{C}(i)$,

(b) *Measurement similarity*: $|x_k - x_j| < b \; \forall k \in \mathcal{C}(i)$.

According to Alg. 1, there are two cases in which node j can be inserted into $\mathcal{C}(i)$: it is inserted by i itself (row 15) or i has inherited it from node $r(i)$ (row 32).

In the first case, in order for j to be inserted in $\mathcal{C}(i)$, it must be $E(i,j) = 1$ (row 10) and $|x_j - x_k| < b \; \forall k \in \mathcal{C}(i)$ (row 14). Therefore, conditions (a) and (b) are both satisfied.

In the second case, it means that $j \in \mathcal{C}(r(i))$ and also that $E(i, r(i)) = 1$, according to the definition of $r(i)$. Furthermore, the update instruction executed before passing the token (row 32) ensures that $\mathcal{C}(i) = \mathcal{C}(r(i))$. Again, the fact that j belongs to the cluster of node $r(i)$ can be due to direct insertion (row 15) or to inheritance from the node that passed the token to $r(i)$ (row 32). In the first of these two cases, it is guaranteed that $E(j, r(i)) = 1$, implying that i, $r(i)$ and j form a path from i to j composed of nodes in $\mathcal{C}(i)$) and that $|x_j - x_k| < b \; \forall k \in \mathcal{C}(r(i)) = \mathcal{C}(i)$: conditions (a) and (b) are both fulfilled. In the latter case, the argument can be repeated iteratively referring to the node that has passed the token to $r(i)$ and so on, tracing the circulation of the token back to the root node in the worst case. ■

***Observation** 1:* It is worth mentioning that the obtained partition is not unique and depends on the scenario conditions such as graph topology, measurements distribution, clustering bound and the randomly chosen root node, nonetheless it results in a set of configurations that are equivalent w.r.t. the given definitions and assumptions. Specifically, if the fault model proposed in Def. 2 is adopted, the presented clustering algorithm allows a correct detection and isolation of faults regardless of the choice of the root node. As an example, Fig. 1 shows the results of the clustering algorithm in a 4–nodes network whose corresponding graph is complete and with equally spaced measurements. It can be seen that the resulting partitions are different according to the choice of the root node and in both cases the clusters properties introduced in Def. 1 are valid. However, this is not an issue for the scope of this work: indeed, if the fault model proposed in Def. 2 is adopted, the presented clustering algorithm allows a correct detection and isolation of faults regardless of the chosen root node, as it will be shown in the following.

C. Cluster-based Consensus and Fault Detection

The proposed clustering strategy can be exploited before the implementation of a consensus procedure, in order to exclude all the nodes that could provide biased information from the measurement averaging. More specifically, the distributed clustering algorithm Alg. 1 can be modified to

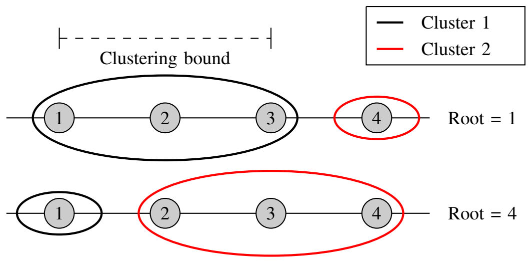


Fig. 1. Linear measurement distribution and clusters obtained with different roots in a 4–nodes complete network.

compute a consensus matrix P based on the Metropolis-Hastings strategy and on a clustering bound b that is suitably set according to the measurements distribution range $\mathcal{R}$.

Furthermore, when the generic node i is selected (via token passing), it computes the consensus weights $P(i,:)$ by exchanging the information on its own and its neighbors degrees. It is worth to notice that the considered degrees are those related only to the cluster's connected subgraph, that is $P(i,j) \neq 0$ if and only if nodes i and j are in the same cluster. The consensus algorithm can then be applied, leading every node i to converge to the mean value of the measurements in the cluster. As a result, if a node j is faulty, the overall procedure:

1) results in node j isolation (because the *measurement similarity* criterion is not verified) *before* the consensus procedure, thus avoiding both the a-posteriori verification of the presence of faulty nodes (and a following phase of iterative estimate recomputation) and the unnecessary exchange of information;
2) preserves the connectivity of the rest of the network, allowing the other nodes to converge to consistent clustered solutions (i.e. subgraphs exhibiting the same average values).

In Fig. 2, it is shown a simple (static) example of average consensus in a connected network made up of $N = 100$ nodes: the quantity to estimate is a temperature of $T = 300K$ and the sensors measurements are affected by additive uniform noise. Faults are generated by a Bernoulli distribution with fault probability $p = 0.4$ and a node is considered faulty if its measurement falls out of a range $\mathcal{R} = [290K,\ 310K]$. The bound for the clustering algorithm is then set to $b = 20K$. The distributed cluster-based consensus procedure is applied, and it can be observed that the estimates of healthy nodes (reported in blue) converge to the true value T, since they all belong to the same cluster. Conversely, faulty nodes (red lines) are either isolated or clustered together separately from the healthy ones, without affecting the estimation procedure. On the other hand, the standard consensus procedure takes into account the estimates of all nodes in the network (healthy and faulty), resulting in a final estimate that is substantially different from the true value T.

***Observation** 2:* Although the algorithm well performs in the general case, an additional remark is due with respect to particular measurements distribution that may lead to *false positives* and/or *false negatives*, i.e. undetected faulty nodes and healthy nodes labeled as faulty respectively. According

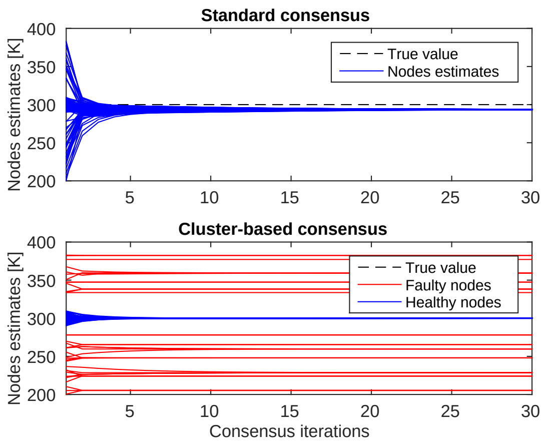


Fig. 2. Standard vs. Cluster-based consensus strategies in a network with faulty nodes.

to Def. 2, the presence of false negatives can be always avoided by setting the value of clustering bound b equal to twice the measurement noise bound, $b = 2\delta$. Indeed, this condition guarantees that healthy nodes are always included in the same cluster (provided that they are part of a connected graph), even in the worst case represented by two nodes whose measurements are positioned at the opposite sides of $\mathcal{R}$, i.e. $x_1 = x - \delta$ and $x_2 = x + \delta$. Unfortunately, this choice may cause the emergence of false positives: nodes whose measurements are in the range $\{(x-3\delta, x-\delta) \cup (x+\delta, x+3\delta)\}$ can be included in the cluster of healthy nodes even if they are faulty. However, statistical considerations, here omitted for brevity, show that such cases occur with low probability (some percent). Numerically, with the same example as before ($N = 100$, $T = 300K$, $p = 0.4$, $b = 20K$) and the realistic assumption that the bias value exceeds 3δ with probability $\alpha = 3/4$, the average number of false positives approximates 5.

IV. Distributed Dynamic Estimation

The proposed algorithm can reveal useful also in a more interesting scenario, namely distributed estimation in a dynamical system: nodes take periodic noisy measurements of a physical quantity that evolves over time and have to provide an on-line estimation of its behavior.

A. Distributed Dynamic Estimation Problem

We consider a network of N nodes that need to estimate a global quantity $T(\cdot) \in \mathbb{R}$ (e.g. a temperature) that changes according to the discrete stochastic model

$$T(s+1) = T(s) + v(s) \qquad (3)$$

where $v(s)$ is a zero-mean random variable with covariance q and s is the sampling instant ranging from 1 to M.

The sensors measure a noisy quantity

$$\mathbf{x}(s) = T(s)\mathbf{1} + \mathbf{w}(s) + \mathbf{f}(s) \qquad (4)$$

where vector $\mathbf{x}(s) \in \mathbb{R}^N$ contains the measurements of the nodes at the s-th sampling instant, the measurement noise vector $\mathbf{w}(s)$ is an independent zero-mean random vector with covariance matrix $R = rI$, and vector $\mathbf{f}(s)$ represents the additive bias provided by faulty sensors. Through $\mathbf{f}(s)$, the model allows to account for different situations, such as:

- *non-faulty node*: $f_i(s) = 0,\ \forall s$;
- *permanent fault*: $f_i(s) = \bar{f},\ \forall s$;
- *sudden fault*: $f_i(s) = \bar{f} \cdot u(s - \bar{s}),\ \bar{s} > 1$;
- *temporary fault*: $f_i(s) = \bar{f} \cdot [u(s - \bar{s}_1) - u(s - \bar{s}_2)]$, $\bar{s}_2 > \bar{s}_1 > 1$;
- *linear degrading fault*: $f_i(s) = \bar{f} \cdot \frac{s}{M-\bar{s}} \cdot u(s-\bar{s}),\ \bar{s} > 1$;

where $\bar{f} \in \mathbb{R}, \bar{f} \neq 0$ and $u(\cdot)$ is the unit step.

The distributed dynamic estimation problem concerns the computation by every node of an optimal estimate $\hat{T}(s)$ of the real value $T(s)$ at each sampling instant. It is well known that the optimal estimator for the evolution of a dynamical system is given by the Kalman filter [19], that can be implemented in a distributed way using average consensus [20]. In this scenario, nodes are able to compute the mean of a set of values only through local communication and the global optimal estimate can be derived by each sensor node at the s-th sample through the following two-step procedure.

1) *Kalman estimation*: each node updates its local estimation according to the steady-state Kalman filter equation

$$\hat{\mathbf{T}}(s) = (1-l^*) \cdot \hat{\mathbf{T}}(s-1) + l^* \cdot \mathbf{x}(s), \quad \hat{\mathbf{T}}(1) = \mathbf{x}(1) \quad (5)$$

2) *Average consensus*: nodes perform average consensus algorithm on their local estimations

$$\hat{\mathbf{T}}(s,\, k+1) = P \cdot \hat{\mathbf{T}}(s,\, k), \quad 0 \leq k < k_{max} \quad (6)$$

where k is the index of consensus iterations and s is the sampling instant, hence the initial condition $\hat{\mathbf{T}}(s,\, 0)$ is the value computed by (5).

The two design parameters in this procedure are the optimal Kalman gain l^* and the consensus matrix P. The P matrix is chosen according to what previously discussed in Sec. II, while the *centralized gain* $l_c^* = \frac{-q+\sqrt{q^2+4q\bar{r}}}{2\bar{r}}$, with $\bar{r} = \frac{r}{N}$, results to be the optimal gain value [20], depending on the measurement and process variances.

B. Cluster-based Consensus in Dynamic Framework

In a network with faulty nodes, the cluster-based consensus algorithm of Subsec. III-C can be implemented at each sampling instant using the local Kalman estimation (5) as the initial value.

This approach guarantees an efficient detection and isolation of faults, however the overall estimation-clustering-consensus procedure adds a relevant computational burden that must fit within a single sampling interval, for a practical implementation. More formally, the following result holds:

Theorem 3: To address the dynamic estimation problem in a network made up of N nodes, some of which are faulty with probability p, the sampling period T_s must satisfy

$$T_s \geq \left[4 + k_{max}\,(1 - p\alpha)^2\right] \cdot N^2 \cdot t_m \quad (7)$$

being α the probability that the bias value exceeds 3δ (i.e. the fault is detectable using the cluster-based approach) and t_m the worst-case information exchange time between two nodes.

Proof: In a N nodes network, the clustering procedure requires at most $4N^2$ communication exchanges (Thm. 1), at the end of which $\bar{N}$ nodes labelled as healthy are selected.

Under the hypothesis of the theorem, the number of (truly) healthy nodes in the network is $N(1-p)$, while the maximum number of false positives can be defined as $Np(1-\alpha)$, being α the probability that a faulty node is not a false positive. As a consequence, $\bar{N} = N(1 - p\alpha) \geq N(1 - p)$ nodes should be considered in the consensus complexity analysis.

Then, in a single consensus iteration, we have at most two exchanges for each edge in the subgraph $\bar{\mathcal{G}} = (\bar{\mathcal{V}}, \bar{\mathcal{E}})$ composed of $\bar{N}$ nodes, yielding an upper bound of $\bar{N}^2$ messages per iteration. Therefore the total number of messages exchanged is bounded by $k_{max}[N(1-p\alpha)]^2$, where k_{max} is the maximum number of iterations (setup parameter).

Summing up the two phases, we can set a maximum bound for the average number of messages that a single node exchanges during the algorithm execution (in the worst case).

Moreover, being t_m the transmission time for a message in the worst-case, we obtain an upper bound for the execution time of the cluster-based consensus algorithm as

$$\left[4 + k_{max}(1 - p\alpha)^2\right] \cdot N^2 \cdot t_m \quad (8)$$

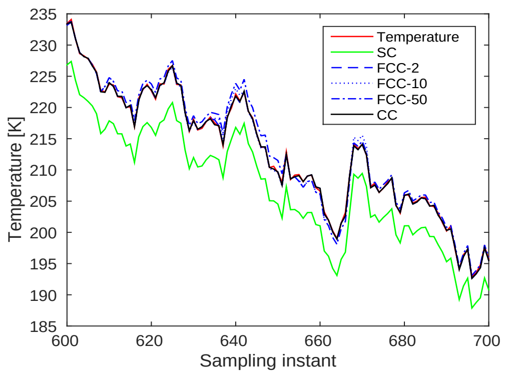
that represents a lower bound for the period T_s. ■

The discussion carried out in the proof is valid in the worst-case of a complete graph topology, whereas, in the general case of a random geometric graph, simulations results highlight that the clustering phase computational burden is often higher than that of the consensus procedure.

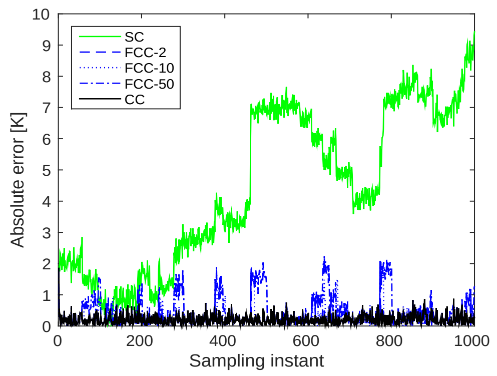
C. Fast Cluster-based Consensus

In order to reduce the impact of the clustering strategy in a dynamic estimation framework in terms of computational and communication costs, a lighter version of the cluster-based consensus algorithm can be devised (*fast cluster-based consensus*), supported by a deeper analysis on the emergence of faults. Specifically, we can observe that when the set of faulty nodes does not change between two consecutive sampling instants, i.e. $\mathbf{f}(s) = \mathbf{f}(s+1)$, the clustering procedure is executed twice but returns the same network partition. To avoid such a redundant behavior, it might be convenient to perform the clustering phase only every $\tilde{s}$ sampling instants thus reducing the average number of messages exchanged but decreasing the overall robustness to faults.

To provide a reference value for this design parameter $\tilde{s}$, let us consider a network of N nodes, each one with probability p of being faulty at a certain sampling instant. The number F of faults at each instant is a binomial variable of parameters N and p, while the number S of samples before having a faulty node is a geometric random variable of parameter $\rho = P[F \geq 1] = 1 - (1-p)^N$. It follows that the parameter $\tilde{s}$ can be set in order to ensure $\tilde{s} \leq \mathbb{E}[S] = \rho^{-1} = (1 - (1-p)^N)^{-1}$.



(a)



(b)

Fig. 3. Dynamic distributed estimation with different consensus algorithms. (a) Detail of estimation evolution. (b) Absolute error evolution.

D. Numerical Validation

To clarify and validate the proposed approach, a numerical example of distributed dynamic estimation in presence of faults is performed on a network of $N = 100$ nodes associated to a random geometric graph. The quantity T to estimate is sampled over $M = 1000$ measurement intervals, its initial value is $T(0) = 300K$ and its covariance is $q = 5$. The measurement noise is Gaussian distributed with covariance $r = 5$ and a bound of $b = 15K$ is set for the clustering algorithm. According to the models presented in Sec. IV-A, fault events and related fault types are randomly generated with probability $p = 5 \cdot 10^{-4}$ at each sampling instant, with $\bar{f}$ being an uniform random variable in the range $\{[-100K, -15K) \cup (+15K, 100K]\}$. For each consensus procedure, $k_{max} = 200$ iterations have been performed.

The results of the dynamic estimation procedure are reported in Fig. 3 for several consensus algorithm versions, namely standard consensus (SC), cluster-based consensus (CC) and fast cluster-based consensus (FCC) with different values of $\tilde{s}$ (2, 10, 50), and these are compared with the true T value (red line). It can be observed that the consensus procedures based on clustering techniques outperform the standard algorithm in terms of estimation accuracy, since the latter is strongly affected by the presence of faults. In particular, the absolute error for the standard consensus algorithm raises over time, up to almost 10K (Fig. 3(b)).

In this example, the differences between cluster-based consensus and its faster variants are almost negligible in terms of estimation accuracy, although the FCC-50 version is such that $\tilde{s} = 50 > \mathbb{E}[S] = 20$. On the other hand, in terms of average number (n_m) of messages exchanged between all nodes at each sampling instant for what concerns the clustering process, the benefit of increasing $\tilde{s}$ are clearly visible. Indeed, given that $n_m = 2697$ when clustering procedure takes place at each sample instant (CC), this value decreases to 1350, 269 and 53 when FCC is adopted with, respectively, $\tilde{s} = 2$, $\tilde{s} = 10$ and $\tilde{s} = 50$.

To further assess the performance offered by different consensus strategies and limit dependency of results from random measure noises and graph topology, a total of 1000 simulations have been executed with the same parameters of the scenario just presented. The strategies efficiency has been evaluated in terms of *average absolute error*, i.e. the mean absolute error computed over all sampling intervals, and *average number of messages* exchanged at each sampling instant for what concerns the clustering process. The results, averaged over all the simulations, are reported in Tab. I.

TABLE I
RESULTS FOR DISTRIBUTED DYNAMIC ESTIMATION AVERAGED OVER 1000 NUMERICAL SIMULATIONS

Strategy	**Average absolute error**	**Average no. messages**
SC	2.729 K	-
CC	0.316 K	2656
FCC ($\tilde{s} = 2$)	0.320 K	1329
FCC ($\tilde{s} = 10$)	0.371 K	265
FCC ($\tilde{s} = 50$)	0.582 K	52

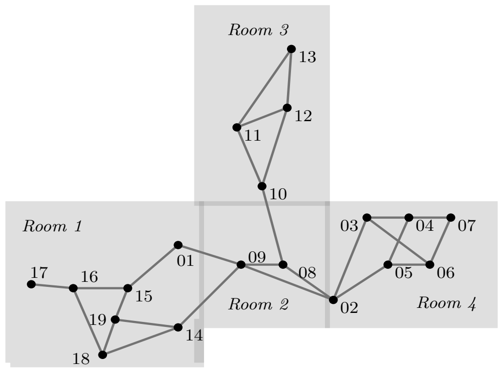
Once again it emerges how the clustering technique strongly increase the robustness of consensus, outperforming the standard procedure in terms of estimation accuracy. The faster versions of the cluster-based consensus algorithm allow to decrease the average number of messages proportionally with the chosen value of $\tilde{s}$: approximately we have $n_m(\tilde{s}) = n_m(1)/\tilde{s}$, where $n_m(1)$ is the average number of messages exchanged if the clustering process is repeated at each sample instant (CC). Such an improvement clearly trades off with a reduction in estimation accuracy. However, even considering a very fast version of the algorithm (FCC-50), the mean absolute error is still more than 4 times smaller than that achieved with standard consensus algorithm.

V. Validation on a Real Scenario Dataset

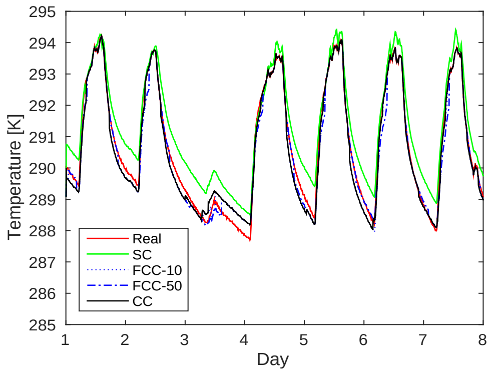
The efficiency of the cluster-based consensus algorithm has been then tested on the data coming from a real distributed estimation scenario. We analyze the temperature evolution in a primary school monitored by $N = 19$ sensors placed in four rooms and connected as in Fig. 4(a), which have collected data with a sampling time of five minutes for four months.

A. Static Cluster-based Consensus

We initially consider a static framework, where the cluster-based consensus algorithm evaluates one single measure-



(a)



(b)

Fig. 4. Real temperature sensor network. (a) Testbed setup. (b) Results of dynamic estimation with different consensus algorithms.

ments set. At the end of the procedure ($k_{max} = 300$), sensors result to be correctly divided into four clusters according to the rooms they belong to. As a consequence, we can state that, imposing a suitable cluster bound ($b = 2K$), the distributed clustering algorithm allows to recognize the topology of a structured environment. Then, non-faulty sensors of each room converge to the average of their measurements, while faults are excluded from consensus computation, as it is the case of a permanently fault node in room 1 (*sensor 01*) that has been correctly isolated and excluded from the consensus computation.

The cluster-based consensus strategy is compared with the standard one in Tab. II, which reports the error between the consensus value and the average of the measurements of the nodes in the same room for a few sample nodes. The cluster-based approach is able to correctly converge to the average of the measurements for each room, while the standard consensus algorithm always provides non zero errors since it is not able to discern among the different rooms temperature profiles or to recognize the presence of the faulty node in room 1.

TABLE II
COMPARISON BETWEEN STANDARD AND CLUSTER-BASED ALGORITHMS FOR STATIC CONSENSUS

Room	Sensor	Consensus value		Error	
		SC	CC	SC	CC
1	no. 16	293.48 K	295.25 K	1.76 K	0 K
2	no. 08	293.48 K	290.44 K	3.05 K	0 K
3	no. 12	293.48 K	292.33 K	1.15 K	0 K
4	no. 04	293.48 K	292.81 K	0.67 K	0 K

B. Dynamic Cluster-based Consensus

The attention is subsequently focused on the temperature evolution during a week comparing the results of different strategies: standard consensus (SC), cluster-based consensus (CC) and fast cluster-based consensus (FCC) with $\tilde{s}$ =10 and $\tilde{s}$ =50. The cluster bound b and the maximum iteration number k_{max} are set to $2K$ and 300 respectively, while the evaluated measurement intervals are $M = 2016$ (samples every 5 minutes).

The results of the dynamic estimation are depicted in Fig. 4(b). For the sake of clarity, only the trend relevant to one node (*sensor 02*) is reported and the real temperature evolution is computed as the mean of the measurements of the nodes in the same room at each sampling instant. The cluster-based approach provides better performance in terms of estimation accuracy, whereas the standard procedure is less precise especially in correspondence to abrupt variations. The more substantial ones are related to the weekend: a rapid decrease of the temperature can be observed between the third and fourth day, i.e. on Saturday and Sunday respectively, during which the heating is switched off. Similarly, for each day is possible to distinguish the daytime and nighttime hours. In all cases, the cluster-based approaches emerge as the best solutions for the dynamic estimation.

VI. CONCLUSIONS AND FUTURE WORKS

In this paper, a distributed cluster-based consensus algorithm has been proposed to deal with the average consensus problem in faulty WSANs. Through this procedure, the nodes in the network are partitioned into clusters according to measurements similarity and nodes connectivity: in doing so, faulty sensors remain isolated and do not concur to the average consensus procedure. This algorithm can also be integrated into a Kalman filtering framework and allows performing robust distributed estimation of a dynamic quantity.

The results obtained in simulations with a large number of nodes and different kind of faults show the good performance of the proposed strategy, also confirmed by the validation on a real scenario dataset. Further developments are currently devoted to a more complete assessment of the procedure also in comparison with other techniques for distributed fault detection in sensor networks.

References

[1] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 138–162, 2007.

[2] C. Chong and S. Kumar, "Sensor networks: evolution, opportunities, and challenges," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247–1256, 2003.

[3] I. F. Akyildiz and I. H. Kasimoglu, "Wireless sensor and actor networks: research challenges," *Ad Hoc Networks*, vol. 2, no. 4, pp. 351–367, 2004.

[4] K. Romer and F. Mattern, "The design space of wireless sensor networks," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 54–61, 2004.

[5] W. Ren and R. Beard, "Consensus seeking in multi-agent systems under dynamically changing interaction topologies," *IEEE Trans. on Automatic Control*, vol. 50, no. 5, pp. 655–661, 2005.

[6] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. on Automatic Control*, vol. 49, pp. 1520–1533, 2004.

[7] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in multi-agent networked systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[8] F. Garin and L. Schenato, "A survey on distributed estimation and control applications using linear consensus algorithms," in *Networked Control Systems*, ser. Lecture Notes in Control and Information Sciences. Springer London, 2010, vol. 406, pp. 75–107.

[9] W. Li and H. Dai, "Cluster-based distributed consensus," *IEEE Transactions on Wireless Communications*, vol. 8, no. 1, pp. 28–31, 2009.

[10] I. Eyal, I. Kedar, and R. Rom, "Distributed data clustering in sensor networks," *Distributed computing*, vol. 24, no. 5, pp. 207–222, 2011.

[11] K. Detroja, R. Gudi, and S. Patwardhan, "A possibilistic clustering approach to novel fault detection and isolation," *Journal of Process Control*, vol. 16, no. 10, pp. 1055–1073, 2006.

[12] G. Gupta and M. Younis, "Fault-tolerant clustering of wireless sensor networks," in *IEEE Wireless Communications and Networking (WCNC 2003)*, vol. 3, 2003, pp. 1579–1584.

[13] X. Zhao, Z. Gao, R. Huang, Z. Wang, and T. Wang, "A fault detection algorithm based on cluster analysis in wireless sensor networks," in *Proc. of the 7th Int. Conf. on Mobile Ad-hoc and Sensor Networks (MSN2011)*, 2011, pp. 354–355.

[14] F. Pasqualetti, A. Bicchi, and F. Bullo, "Distributed intrusion detection for secure consensus computations," in *Proc. of the 46th IEEE Conf. on Decision and Control (CDC2007)*, 2007, pp. 5594–5599.

[15] K. P. Kihlstrom, L. E. Moser, and P. M. Melliar-Smith, "Solving consensus in a byzantine environment using an unreliable fault detector," in *Proc. of the Int. Conf. on Principles of Distributed Systems (OPODIS)*, 1997, pp. 61–75.

[16] F. Pasqualetti, A. Bicchi, and F. Bullo, "Consensus computation in unreliable networks: A system theoretic approach," *IEEE Trans. on Automatic Control*, vol. 57, no. 1, pp. 90–104, 2012.

[17] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[18] J. C. Laprie, "Dependable computing and fault tolerance: concepts and terminology," in *Proc. of 15th Int. Symp. on Fault-Tolerant Computing (FTSC-15)*, 1985, pp. 2–11.

[19] B. Anderson and J. Moore, *Optimal Filtering*. Prentice Hall, 1979.

[20] R. Carli, A. Chiuso, L. Schenato, and S. Zampieri, "Distributed kalman filtering based on consensus strategies," *IEEE Journal on Selected Areas in Communications*, vol. 26, pp. 622–633, 2008.