International Journal of Innovative Computing, Information and Control
Volume **9**, Number **3**, March **2013**

ICIC International ©2013 ISSN 1349-4198
pp. 915–938

COLLABORATIVE DIAGNOSIS AND COMPENSATION OF MISBEHAVING NODES IN ACYCLIC CONSENSUS NETWORKS: ANALYSIS AND ALGORITHMS

Gianfranco Parlangeli

Dipartimento di Ingegneria dell'Innovazione
Universitá del Salento
via per Monteroni, Lecce, Italy
gianfranco.parlangeli@unisalento.it

Received July 2011; revised January 2012

Abstract. *In this paper the problem of intrusion diagnosis and compensation for a collaborative networked system with acyclic communication graph is considered. The main novel contributions of the paper are two: (a – monitoring set selection) provide necessary and sufficient conditions for the selection of a subset of monitoring nodes and (b – diagnostic and compensation algorithms) provide a diagnostic algorithm to select, exclude and compensate for a misbehaving node within the network based on a collaboration between the chosen monitoring nodes. The proposed solution has a collaborative multinode architecture with precise and easy instructions for each node of the monitoring network. Computational issues are investigated and the adaptation of the diagnostic algorithm in the presence of bounded uncertainties is analyzed. Simulations are made in order to validate theoretical results.*

Keywords: Fault diagnosis and compensation, Multi-agent systems, Consensus networks

1. **Introduction.** In the last decade a significant thrust of research has been devoted to the analysis of dynamical systems over graphs and their applications in robotics and distributed estimation. An important class of problems within this area is the agreement or consensus problem (see, e.g., [1, 2, 3] and references therein). In view of the Motivations reported below, the topic of this paper is a research on collaborative strategies between a selected subset of nodes in order to attain the ability of an automatic reaction to an intrusion/fault, as stated more formally in the next Objective of the Research and Problem Statement sections.

1.1. **Motivations.** An important issue of such systems is the standing assumption that each agent behaves according to the consensus protocol. This is a venturous assumption because, if any of the agents injects exogenous values (maliciously or as a consequence of a fault), then this agent can drive the evolution of the group arbitrarily.

This is a classical problem in Computer Science [4] and it is becoming more and more popular in many other application areas of multi-agent systems. A fundamental peculiarity of the solution for the classical problem is that a single node performs the intrusion detection task based on locally collected data only. A direct consequence of this approach is that a node running a diagnostic algorithm should have sufficient logical redundancy of its local data to infer the presence of an intruder, and this implies a condition on the network vertex connectivity [4, 5, 6]. One straight consequence of these results is that, if the communication network has connectivity less than 3, then any single node cannot infer from local data the presence of a single misbehaving node in the network. This

result has a strong negative impact for low-connectivity networks. Indeed, it proves that monitoring the network from any single node is useless because any single misbehaving node can drive a consensus network without being detected.

On the other hand, low-connectivity communication graphs are useful in various applications. From a practical point of view, as claimed in [7], 'low-connectivity and full-coverage WSNs have many real-world applications'. Indeed, such graphs can be associated to the minimal amount of power consumption for each node in WSN applications or to the maximal spreading of mobile agents for multi-robot applications. Energy consumption is one major issue of WSN [8], so these networks are often designed with sparse graphs.

1.2. **Literature review.** As far as the consensus problem is concerned, the basic problem is the task of reaching a consensus on a common value of the desired quantity by performing local computation and exchanging local information [2, 3]. Many important applications in multi-agent systems or distributed computing can be reduced to a consensus problem, such as network synchronization [9], robot rendez-vous [10] and many others [11].

The problem of intrusion detection is a classical issue in the area of distributed computing [4, 12] and in the last years this problem has been considered in the control systems community [5, 6]. In these papers, a single node makes all the elaboration and takes the decision on the basis of its own local data, and these results do not apply for low-connectivity networks. Fault diagnosis and compensation applications in the framework of multi-agent and distributed systems have been recently studied in [13, 14].

Recently, research on distributed and cooperative IDS is an active research field in the mobile ad-hoc networks community and it is recognized as a major challenge by several authors [15, 16, 17]. This research objective is becoming more and more popular also for Internet (see, e.g., [18, 19]) using peer-to-peer architectures.

We now review some recent results on the importance of low-connectivity networks for a variety of practical applications. In [20], the importance of sparse network deployments in multi-agent underwater missions is stressed and this, associated to the unaccessibility of the environment, calls for the design of 'disruption-tolerant networks (DTNs)'. The problem of Maritime Communications System has also been considered in [21]. Low-connectivity network design is a fundamental problem also in the practical design of telecommunication networks [22]. In [22], the author focuses on the design of graphs with node degree less or equal than two. Finally, acyclic graphs have been recently considered for a study on controllability [23].

1.3. **Objective of the research.** In view of the important applications of low-connectivity networks, the main motivation of this paper is a research on collaborative multinode detection systems. The most ambitious objective of this work is to overcome the limitations of the small amount of redundancy available to every single node of a low-connectivity network by the use of collaborative multinode diagnostic strategies. This is an important step toward novel solutions of practical interest for the Intrusion Detection Problem in modern networks [15, 16, 17].

In this paper we start this investigation considering a special class of 1-connected graphs, namely acyclic graphs. The main objectives of the paper are two: (a) to deduce necessary and sufficient conditions to select a subset of nodes in order to collect sufficient information on system evolution to unambiguously infer the presence of a misbehaving node and (b) to find a diagnostic algorithm to select and exclude the misbehaving node using the minimal amount of shared data. Furthermore, a compensation of the effects of the intruder on the objective function is performed to restore an unbiased consensus value. Some preliminary

results on point (a) relative to a restricted class of the systems here considered, namely nearest-neighbor average consensus networks, are presented in [24].

Though the setting of this paper is rather theoretical, one of the most important goals is to provide guidelines for collaborative IDS design that can be easily (and effectively) implemented in practice. We discuss on the ease and convenience of its use, especially focusing on: (a) the distinction between the off-line and on-line computation, and (b) the difference between local data/computation and data/computation to share between monitoring nodes (a goal is to minimize the amount of shared data). An additional goal is to derive algorithms requiring computationally low on-line elaboration because they should be run by nodes with reduced computational capability in practical applications.

1.4. **Notation and preliminaries from graph theory.** The paper is organized as follows. In Section 2, the basics of consensus networks and some definitions are given in order to set the problem properly. Section 3 is the main section on node selection. After some technical algebraic results, easy instructions on how to choose a complete set of monitoring nodes are given. Section 4 is the main section on diagnostic algorithms. In this section the algorithms for detection and isolation are thoroughly investigated and a decentralized, directly implementable algorithm is given at the end of each subsection. Section 5 describes the compensation of the bias induced by the misbehaving node. In Section 6, some implementation issues are discussed and simulation results are shown. Section 7 concludes the paper with a brief summary and some considerations on future research activity on this topic.

In the following, we denote 0_d the vector of dimension d with zero components and $0_{d_1 \times d_2}$ the matrix with d_1 rows and d_2 columns with zero entries. Given a vector $v \in \mathbb{R}^n$, we denote with $(v)_\ell$ the ℓ-th component of v. I_n represents the $n \times n$ identity matrix, finally, $\mathbf{e}_i$, $i \in \mathbb{N}$, denotes the i-th element of the canonical basis, e.g., $\mathbf{e}_1 = [1\ 0\ \dots\ 0]^T$. Vector $\mathbf{1}$ is the vector (of suitable dimension) with each component equal to 1.

A brief summary of notations and definitions of graph theory follows. The interested reader can consider books in graph theory for details (e.g., [25, 26]).

A graph $\mathcal{G}$ is an ordered pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \dots n\}$ is the set of vertices and $\mathcal{E} = \{(i,j) | i \text{ is connected to } j\}$ represents an edge set. Two vertices i and j such that $(i,j) \in \mathcal{E}$ are said to be *adjacent* or *neighbors*; the set of neighbors of a node i is denoted $\mathcal{N}_i$ (i.e., $\mathcal{N}_i = \{j \in \mathcal{V} : (i,j) \in \mathcal{E}\}$) and the number of neighbors $|\mathcal{N}_i|$ is called *degree* of node i and it is also denoted with d_i; the degree matrix $\mathcal{D}$ is a diagonal matrix whose diagonal entries are each node degree $\mathcal{D} = \text{diag}\{d_i\}$. An effective algebraic tool representing the elements of $\mathcal{E}$ is the adjacency matrix $\mathcal{A}$ whose (boolean) entries are $[\mathcal{A}]_{ij} = 1$ if $(i,j) \in \mathcal{E}$, 0 otherwise.

We introduce some special graphs that will be of interest in the rest of the paper. A *tree* is a graph in which any two vertices are connected by exactly one simple path. The vertices of a tree having degree one are called *leaves*. A *path graph* is a special tree containing only nodes of degree two except for the (two) leaves of degree one. For the sake of brevity, we will denote a path graph of length n with P_n (and labels 1 and n refer to the two leaves).

A cycle of a graph is a subset of the edge set that forms a path such that the first node of the path corresponds to the last. A graph containing no cycles of any length is known as an acyclic graph, whereas a graph containing at least one cycle is called a cyclic graph. A connected acyclic graph is a tree, and a disconnected acyclic graph is called a forest.

Considering a tree graph, every two nodes i and j can be connected by exactly one path; we call the *distance* between i and j the number of edges that constitute this path

(independently from the weight associated to each edge) and it is denoted as $d(i,j)$ in the following.

The eccentricity of a vertex v is the greatest distance between v and any other vertex $e_v = \max_{u\in\mathcal{V}} d(v,u)$. It can be thought of how far a node is from the most distant node in the graph. The *radius* of a graph is the minimum eccentricity of a graph, and the *diameter* of a graph is the maximum eccentricity of the graph, i.e., it is the greatest distance between any pair of vertices.

2. **Collaborative Networked Systems: Dynamic Model.** Following the mathematical set-up described in [2], in this work we consider a group of n discrete-time agents following the dynamics $x_i(t+1) = x_i(t) + u_i(t)$. Each agent can communicate with a subset of other agents (its *neighbors*). The evolution of the whole group can be effectively described using a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the vertex set $\mathcal{V} = \{1, \dots, n\}$ is associated to a labeling of each single agent x_i and the edge set $\mathcal{E} = \{(i,j) | i \text{ can communicate with } j\}$ describes compactly the communication between agents.

Adopting the notation of [23], each collaborative agent chooses its input according to a general distributed consensus algorithm:

$$u_i(t) = \sum_{j\in\mathcal{N}_i} l_{ij}(x_j(t) - x_i(t)). \tag{1}$$

If each agent behaves according to (1), then the evolution of the group can be compactly written using an aggregated state $\mathbf{x} = [x_1\ x_2\ \dots x_n]^T$:

$$\mathbf{x}(t+1) = A\mathbf{x}(t), \quad A = (I - \tilde{L}) \tag{2}$$

where $\tilde{L}$ is called the *generalized Laplacian* (or *weighted Laplacian*) of the graph G:

$$(\tilde{L})_{ij} = \begin{cases} -l_{ij}, & \text{if } i \neq j \\ \sum_{j=1, j\neq i}^{n} l_{ij}, & \text{if } i = j \end{cases} \tag{3}$$

where $l_{ij} > 0$ is a positive weight if $(i,j) \in E$ and $l_{ij} = 0$ if $(i,j) \notin E$.

The state matrix $A = (I - \tilde{L})$ in (2) is a structured matrix with some interesting properties, e.g., if each row-sum of the non-diagonal weights is less than 1 (i.e., $\max_i(\sum_{j\neq i} l_{ij}) < 1$) and if the graph is connected then it is a matrix with a simple eigenvalue in 1 and all the other eigenvalues are inside the unit circle. In the following we assume that all weights are chosen according to this rule. A first important consequence of it is that, if every node fairly chooses its input according to (1), system evolution asymptotically converges (or, equivalently, *reaches a consensus*) to a linear function of the initial state $f(\mathbf{x}_0) = \dfrac{\gamma^T \mathbf{x}_0}{\gamma^T \mathbf{1}}$ where γ is the eigenvector of A^T corresponding to the eigenvalue 1 [2].

A specific function of wide interest for its practical applications is $f(\mathbf{x}_0) = \dfrac{\sum_{i=1}^{n} (\mathbf{x}_0)_i}{n}$; the necessary and sufficient condition to achieve a consensus on this value is to choose weights l_{ij} in (1) such that $\sum_{j\neq i} l_{ij} = \sum_{j\neq i} l_{ji}$ for all i, j. We will refer to this framework as the *average consensus*.

Consider now the presence of an agent who does not behave according to the control law (1); this can be modeled by

$$u_i(t) = \sum_{j\in\mathcal{N}_i} -l_{ij}(x_i(t) - x_j(t)) + \bar{u}_i(t) \tag{4}$$

for some nonzero $\bar{u}_i(t)$. We call such node a *misbehaving node*.

The impact of the presence of an intruder even for few instants can be dramatic on the control objectives. If the intruder action is persistent then consensus cannot be reached, in any case (even for a single impulse of exogenous injection) the consensus value deviates from $f(\mathbf{x}_0) = \dfrac{\gamma^T \mathbf{x}_0}{\gamma^T \mathbf{1}}$ to

$$f(\mathbf{x}_0, \bar{u}(\cdot)) = \frac{\gamma^T \mathbf{x}_0 + (\gamma)_j \sum_{\tau s.t. \bar{u}(\tau) \neq 0} \bar{u}(\tau)}{\gamma^T \mathbf{1}} \tag{5}$$

($f(\mathbf{x}_0, \bar{u}(\cdot))) = \dfrac{\sum_{\tau=1}^{n} (\mathbf{x}_0)_i + \sum_{\tau s.t. \bar{u}(\tau) \neq 0} \bar{u}(\tau)}{n}$ in the special case of average consensus). An important straight consequence of (5) is that, even if an intruder is located and excluded, a bias on the final value of the objective function is persistent.

The objective of this work is to design a multinode algorithm to counteract against this phenomenon. In the development of the algorithm, we distinguish between *local elaboration*, i.e., all the elaboration that each monitoring node can do using its own data, and *global information* or *shared information*. Our goal is to design an effective diagnostic elaboration using the minimal amount of global information. The technology for the communication between monitoring nodes is not investigated in the present paper. This aspect is in fact more pertinent to a specific application. However, the focus of the paper is to deduce the minimal amount of information to share in order to have a complete diagnosis.

We assume that a set of nodes $\mathcal{I}_o = \{i_{o_1}, i_{o_2}, \ldots, i_{o_s}\}$ can share information for diagnostic purposes. We call this set the *set of monitoring nodes*. Each monitoring node can access to its own value at each time instant, $y_1(t) = x_{i_{o_1}}(t) = \mathbf{e}_{i_{o_1}}^T \mathbf{x}(t)$, $y_2(t) = x_{i_{o_2}}(t) = \mathbf{e}_{i_{o_2}}^T \mathbf{x}(t), \ldots y_s(t) = x_{i_{o_s}}(t) = \mathbf{e}_{i_{o_s}}^T \mathbf{x}(t)$. Thus, from a mathematical point of view the problem is to infer the presence of some nonzero exogenous $u_i(t)$ from data $y_1(t), y_2(t), \ldots, y_s(t)$. More formally, given the structured state space model:

$$\mathbf{x}(t+1) = A\mathbf{x}(t) + Bu_j(t), \quad A = (I - \tilde{L}), \quad B = \mathbf{e}_j \tag{6}$$

$$y(t) = C\mathbf{x}(t) \qquad C = \begin{bmatrix} \mathbf{e}_{i_{o_1}}^T \\ \vdots \\ \mathbf{e}_{i_{o_s}}^T \end{bmatrix} \tag{7}$$

decide if the evolution $y(t)$ of the system is driven by a nonzero $u_j(\tau)$, $\tau \leq t$ based on the sequence of data $y(0), \ldots, y(t)$ without knowing $\mathbf{x}(0)$, $u_j(t)$ and j (and hence B).

2.1. **Diagnosis of a misbehaving node: intrusion detection, isolation, identification.** In this subsection we introduce definitions and concepts of the classical fault diagnosis literature, ranging from computer science to process automation, to properly state the problem.

- *Intrusion detection* is the ability of the diagnostic system to infer from data the presence of an intruder within the network. Usually the output of a detection system is a boolean signal $\sigma \in \{\texttt{yes}, \texttt{no}\}$. A related important parameter is the intrusion detection time, t_d, that is the time period elapsed from the first time the intruder injects a nonzero input to the detection instant. It expresses how long an intruder can corrupt data within the network before an alert can be given.
- *Intrusion isolation* or *intruder localization* is the capability of revealing the index of the intruder node. This is a step of great importance in the area of cooperative multi-agent systems or distributed computing because it allows the exclusion of the intruder from collaboration.

- *Intrusion identification* is the estimation of the non-null values injected by the intruder. This step is not strictly necessary in an intrusion diagnostic system (the isolation of the intruder allows the exclusion of the misbehaving node from the collaborative scenario) but it is useful to data correction without restarting the system or, in general, to counteract against the intrusion effects.

Problem Statement: Given a networked system described by Equations (6) and (7) potentially subject to the presence of an intruder, the problem we aim at solving is threefold:

- *(Monitoring node selection)* Find conditions to select a suitable set of *monitoring nodes* $\mathcal{I}_o = \{i_{o_1}, i_{o_2}, \dots, i_{o_s}\}$ such that the problem of intrusion detection and isolation has a unique solution for any input $u_j(t)$.
- *(Fault diagnostic algorithms)* Find collaborative operations between the chosen *monitoring nodes* to effectively perform a complete diagnosis. It is assumed that the elements of the set of monitoring nodes $\mathcal{I}_o = \{i_{o_1}, i_{o_2}, \dots, i_{o_s}\}$ can share diagnostic information between themselves.
- *(Fault compensation action)* Find a decentralized collaborative fault compensation strategy from a subset of monitoring nodes to effectively react against the effects of the fault on system performances.

3. **Intrusion Detection System Design: Selection of Monitoring Nodes.** In this section we make a theoretical investigation aimed at obtaining clear guidelines to properly select the set of monitoring nodes. The goal of this investigation is to find an appropriate subset of nodes such that the evolution of such nodes consequent to any nonzero input cannot coincide with any free response of the system. From a theoretical perspective, an appropriate set of monitoring nodes must satisfy this condition because only under this condition the presence of an intruder leaves a unambiguous 'footprint' (the so called 'fault signature') on the data available for diagnosis. So, only under this condition it is possible to find algorithms useful to extract diagnostic information from such data. This condition has been widely exploited for a single-node monitoring in [5, 6]. An effective tool to cope with the intrusion diagnosis is the *strong observability*.

Definition 3.1. [27] *A strictly proper LTI system $\Sigma = (A, B, C)$ described by the equations $\mathbf{x}(t+1) = A\mathbf{x}(t) + Bu(t)$, $y(t) = C\mathbf{x}(t)$ is* strongly observable *if $y(t) = 0\ \forall t > 0$ implies $\mathbf{x}(t) = 0\ \forall t > 0$.*

We now report a statement resuming the main properties of strongly observable systems (see [5, 27]).

Theorem 3.1. *The following statements are equivalent:*

- $\operatorname{rank}\begin{pmatrix} C & 0 & \dots & 0 \\ CA & CB & & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{n-1} & CA^{n-2}B & \dots & CB \end{pmatrix} = n + \operatorname{rank}\begin{pmatrix} CB & & 0 \\ \vdots & \ddots & \vdots \\ CA^{n-2}B & \dots & CB \end{pmatrix}$, *i.e., the column space of the observability matrix is full and linearly independent from the forced response matrix.*
- *System $\Sigma = (A, B, C)$ has no finite invariant zeros, i.e., system matrix*
$$P(z) = \begin{bmatrix} zI - A & -B \\ C & 0_{p\times m} \end{bmatrix}$$
has full rank $\forall z \in \mathbb{C}$.
- *System $\Sigma = (A, B, C)$ is strongly observable.*

The proof of the main result of this section is quite involved. A first step toward this goal is to fix a precise vertex labeling rule:

1. Start labeling from any two leaves of the tree (taking one of them as starting node and the other as ending node), thus labeling trimly one path (say, P_1) of the tree.
2. If there are unlabeled nodes (i.e., if the graph $\mathcal{G}$ is larger than the path P_1), continue labeling a second path (say P_2) from a starting node adjacent to P_1 to an ending node on any leaf not already labeled. Repeat for all paths (say, $P_3, \dots, P_{p1}$) that have one ending point on P_1.
3. If there are other unlabeled nodes, then continue labeling according to point (2) applied to $P_2, \dots, P_{p1}$ instead of P_1. Follow on until all nodes are labeled.

Just for example, consider the graph of the Simulation Results (Figure 3) where we applied the above procedure. We started labeling from the left, thus we labeled the path joining two leaves (nodes 1 and 7 in Figure 3) of the tree according to the indications in point 1. Then, according to point 2, we labeled a second path from node 8 to 11 and finally a third path from 12 to 15.

The above labeling procedure is useful to exploit the structure of the graphs of interest in the paper. According to the labeling rule described above, the (polynomial) matrix $zI - A$ has the structure

$$zI - A = \begin{bmatrix} \mathbf{P_1}(z) & \Lambda_{12} & \dots & \Lambda_{1,p1} & 0 & \dots \\ \Lambda_{21} & \mathbf{P_2}(z) & 0 \dots & 0 & \Lambda_{2,p1+1} & \dots \\ \vdots & \ddots & & & & \\ \Lambda_{p1,1} & 0 & & & & \\ 0 & \Lambda_{p1+1,2} & & & & \\ \vdots & \vdots & & & & \end{bmatrix}, \tag{8}$$

where each $\mathbf{P_i}(z)$ is a tridiagonal polynomial matrix with structure:

$$\mathbf{P_i}(z) = \begin{bmatrix} z - l_{i_1,i_1} & l_{i_1+1,i_1} & 0 & \dots & 0 \\ l_{i_1,i_1+1} & z - l_{i_1+1,i_1+1} & l_{i_1+2,i_1+1} & \dots & 0 \\ \vdots & \ddots & \ddots & & \\ & & & & l_{i_1+n_i,i_1-1+n_i} \\ 0 & & & l_{i_1-1+n_i,i_1+n_i} & z - l_{i_1+n_i,i_1+n_i} \end{bmatrix}$$

and $\Lambda_{\ell k}$, representing the connection between two adjacent paths P_ℓ and P_k, is equal to $\Lambda_{\ell k} = l_{\ell k}\mathbf{e_1}\mathbf{e}_{j_x}^T$, j_x being the position of the junction node in P_ℓ. The presence of vector $\mathbf{e}_1$ is a consequence of the labeling procedure, because the first node of an added path is always the next label of the last node of the previous path. Finally, notice also that $\frac{1}{l_{\ell k}}\Lambda_{\ell k} = \frac{1}{l_{k\ell}}\Lambda_{k\ell}^T$ (i.e., $\Lambda_{\ell k}$ and $\Lambda_{k\ell}^T$ have zero entries in the same positions).

The class of graphs here considered, namely connected acyclic graphs, is easily captured with this labeling. Indeed, any column block of the upper part has only one nonzero Λ submatrix and the other blocks of any column are zero (equivalently, each row-block in the lower part has only one nonzero element).

We split the main result of this section into two for the sake of readability. Here a technical lemma proves that if a monitoring set includes all the leaves then no undetected misbehaving node can act inside the network.

Lemma 3.1. *Given a distributed collaborative multi-agent system described by Equation (2), a misbehaving node j is detectable for any input **if** all the leaves are nodes of the monitoring set.*

Proof: We study the strong observability of the system assuming without loss of generality that a misbehaving node j lies along P_1. According to Theorem 3.1, we directly compute system zeros when all the monitoring nodes are the leaves of the communication graph. Consider

$$P(z) = \begin{bmatrix} zI - A & -\mathbf{e}_j \\ \begin{bmatrix} \mathbf{e}_{i_{o_1}}^T \\ \vdots \\ \mathbf{e}_{i_{o_s}}^T \end{bmatrix} & 0_{p\times m} \end{bmatrix}$$

so $P(z)$ has full rank if and only if its columns are linearly independent $\forall z \in \mathbb{C}$. According to the structure of $zI-A$ described in (8) and considering that the columns corresponding to the 1's of the output or input matrices are surely linearly independent from the others, then it is easy to see that $\text{rank}[P(z)] = 1 + s + \text{rank}[\tilde{P}(z)]$ $\forall z \in \mathbb{C}$ where

$$\tilde{P}(z) = \begin{bmatrix} & 0 & & & & \\ \overline{\mathbf{P}}_{\mathbf{1,j-1}}(z) & \vdots & 0 & \tilde{\Lambda}_{12} & \dots & 0 \\ & 0 & & & & \\ & l_{j,j-1} & & & & \\ & l_{j,j+1} & & & & \\ & 0 & & & & \\ 0 & \vdots & \underline{\mathbf{P}}_{\mathbf{1,n-j}}(z) & \hat{\Lambda}_{12} & & \dots \\ & 0 & & & & \\ \tilde{\Lambda}_{21} & & \hat{\Lambda}_{21} & \underline{\mathbf{P}}_{\mathbf{2}}(z) & & \\ \tilde{\Lambda}_{31} & & \hat{\Lambda}_{31} & 0 & \underline{\mathbf{P}}_{\mathbf{3}}(z) & \\ \vdots & & \vdots & & \ddots & \\ & & & & & \underline{\mathbf{P}}_{\mathbf{s}}(z) \end{bmatrix},$$

$\overline{\mathbf{P}}_{\mathbf{1,j-1}}(z)$ (respectively $\underline{\mathbf{P}}_{\mathbf{k}}(z)$) is matrix $\mathbf{P}_1(z)$ after deletion of the first column (resp., the matrix $\mathbf{P}_{\mathbf{k}}(z)$ after deletion of its last column), i.e.,

$$\overline{\mathbf{P}}_{\mathbf{1,j-1}}(z) = \begin{bmatrix} l_{12} & 0 & \dots & \\ z - l_{22} & l_{23} & \dots & \\ l_{32} & \ddots & \ddots & \\ \vdots & & & \\ 0 \dots & & & z - l_{j-1,j-1} \end{bmatrix}, \ \underline{\mathbf{P}}_{\mathbf{k}}(z) = \begin{bmatrix} z - l_{k_1,k_1} & l_{k_1+1,k_1} & 0 & \dots \\ l_{k_1,k_1+1} & & & \dots \\ 0 & & \ddots & \\ \vdots & & & z - l_{i_{n_i}-1} \\ & & & l_{i_{n_i}} \end{bmatrix}$$

and $\hat{\Lambda}_{12}$ $\tilde{\Lambda}_{21}$ are obtained splitting the Λ_{12} matrix along the j-th row (notice that only one between $\hat{\Lambda}_{12}$ or $\tilde{\Lambda}_{21}$ can be nonzero, they can be both the zero matrix if the misbehaving node j is a junction node).

In order to prove the full rank of matrix $\tilde{P}(z)$, we show that the only linear combination of its columns producing the zero vector is the zero combination, thus proving their linear independence.

Consider a vector v partitioned according to the partition of $\tilde{P}(z)$ (i.e., $v = [v_1\ v_2\ \dots\ v_s]^T$, where the dimension of each v_i equals the dimension of each $\underline{\mathbf{P}}_{\mathbf{i}}(z)$). In order to impose $\tilde{P}(z)v = 0$, we start from the last row-block $\Lambda_{\ell s}v_\ell + \underline{\mathbf{P}}_{\mathbf{s}}(z)v_s = 0$. Since $\Lambda_{\ell s}v_\ell = l_{\ell s}\mathbf{e}_1\mathbf{e}_{jx}^T v_\ell = (l_{\ell s}(v_\ell)_{jx})\mathbf{e}_1$, ℓ being the labeling of the (unique) path joined to P_s, the

previous relation can be put into a more compact form:

$$\begin{bmatrix} l_{\ell s}\mathbf{e}_1 & \underline{\mathbf{P}}_{\mathbf{s}}(z) \end{bmatrix} \begin{bmatrix} (v_\ell)_{jx} \\ v_s \end{bmatrix} = \begin{bmatrix} l_{\ell s} & z-l_{s,s} & l_{s,s+1} & 0 & \dots \\ 0 & l_{s+1,s} & z-l_{s+1,s+1} & l_{s+1,s+2} & 0\dots \\ & 0 & & & \\ & 0 & \ddots & & \\ & \vdots & & & z-l_{s_{n_s},s_{n_s}-1} \\ & & & & l_{s_{n_s},s_{n_s}} \end{bmatrix} \begin{bmatrix} (v_\ell)_{jx} \\ v_s \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \tag{9}$$

and it is easily seen that the matrix on the left is a unimodular square triangular matrix with all diagonal entries equal to nonzero constant entries, so it is always nonsingular. Thus the only solution is $\begin{bmatrix} (v_\ell)_{jx} \\ v_s \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$ for any $z \in \mathbb{C}$.

Going backwards by induction, focusing on a general k-th row-block and considering that v_s, v_{s-1}, …, v_{k+1} are zero vectors of suitable dimensions, one still finds an equation that is formally equal to (9) (because the Λ matrices on the right of $\underline{\mathbf{P}}_\ell(z)$ multiply zero coefficients and there is at most one nonzero Λ on the left of $\underline{\mathbf{P}}_\ell(z)$ as a consequence of the acyclic structure of the graph).

Last equation to consider is $\begin{bmatrix} & 0 \\ \overline{\mathbf{P}}_{\mathbf{1,j-1}}(z) & \vdots \\ & 0 \\ & l_{j,j-1} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$ and following the above argumentations it is easily seen that the matrix on the left is a unimodular square triangular matrix with all diagonal entries equal to nonzero constants, so it is always nonsingular. Thus the only set of column combinators of $\tilde{P}(z)$ to have the zero vector is the zero combination, this implying that the system $(A, \mathbf{e}_j, \begin{bmatrix} \mathbf{e}_{i_{o_1}}^T \\ \vdots \\ \mathbf{e}_{i_{o_s}}^T \end{bmatrix})$ has no finite zeros for any $j \in \{1, \dots, n\}$.

Finally, we prove that this condition is also necessary, thus proving the equivalence (necessary and sufficient) condition in the statement.

Theorem 3.2. *A system running a distributed cooperative control described by Equations (1)-(3) subject to the presence of a misbehaving node (4) and with a set of monitoring nodes $\mathcal{I}_o = \{i_{o_1}, i_{o_2}, \dots, i_{o_s}\}$ has no finite invariant zeros* ***if and only if*** *all the leaves of the communication graph are nodes of the monitoring set.*

Proof: According to Lemma 3.1 if all the leaves are monitoring nodes, then the system has no finite zeros. We now show that if a leaf node (say, i_1) is not a monitoring node then there are finite zeros.

Suppose without loss of generality that i_1 is a leaf on a path, say P_1, of the tree and it is not a monitoring node. Take now the intruder node equal to the a monitoring node on P_1 ($j = i_{o_1}$) if there are a monitoring nodes on P_1, a junction node of P_1 otherwise. Following the same lines of the proof of Lemma 3.1, the s columns corresponding to the 1's of the last rows are linearly independent.

$$\operatorname{rank}(P(z)) = s + 1 + \operatorname{rank}\left(\begin{bmatrix} \mathbf{P_1}(z) & 0 \\ 0 & \mathbf{P_2}(z) \end{bmatrix} \right),$$

where $\mathbf{P_1}(z) = \begin{bmatrix} z-l_{11} & l_{12} & \dots & 0 \\ l_{21} & z-l_{22} & & 0 \\ \vdots & & & l_{j-2,j-1} \\ 0 & & l_{j-1,j-2} & z-l_{j-1,j-1} \end{bmatrix}$ and $\mathbf{P_2}(z)$ is a suitable polynomial matrix.

Now, matrix $\mathbf{P_1}(z) \in \mathbb{R}[z]^{(j-1)\times(j-1)}$ has the form of $\mathbf{P_1}(z) = zI - P_1$, P_1 a real constant square matrix, and so it is rank deficient for all $z^* \in \mathbb{C}$ such that $\det[\mathbf{P_1}(z^*)] = 0$, i.e., for all the eigenvalues matrix P_1. This implies that at least $j-1$ invariant zeros are present, thus proving the statement.

Remark 3.1. *Notice that, even if the result of Theorem 3.2 is rather technical, its consequences have a strong practical importance. It basically states that a subset of nodes collects sufficiently rich data on system evolution to infer the presence of an intruder if and only if it contains all the leaves of the graph.*

4. **Design of the Diagnostic Algorithms.** In this section the basics of the proposed diagnostic algorithm are posed. First notice that an equivalent tool to describe the dynamic relation between the intruder and each monitoring node is the an *Auto Regressive Moving Average* (ARMA) model:

$$\begin{aligned} AR(y_\ell(t)) &= MA_j(u(t)) \qquad \text{where} \\ AR(y_\ell(t)) &= y_\ell(t) + a_1 y_\ell(t-1) + \dots + a_n y_\ell(t-n) \\ MA_j(u(t)) &= b_1 u(t-\delta_{\ell j}) + b_2 u(t-\delta_{\ell j}-1) + \dots \end{aligned} \tag{10}$$

where the delay between a nonzero input and the corresponding output $\delta_{\ell j}$ only depends on the distance between the two nodes $\delta_{\ell j} = d(\ell, j) + 1$ while weights in (1) $l_{i,j}$ are related with the coefficients a_i, b_i.

There is a strict connection between the ARMA model (10) and the state space model (6). By example, coefficients a_i, b_i in (10) are the same of the coefficients of the polynomials of the transfer function. Finally, notice that the structure of the $AR(\cdot)$ elaboration (i.e., its coefficients) does not depend on the specific input or output node and that the $AR(\cdot)$ elaboration requires n data, so it can be computed at $t \geq n$.

We briefly summarize the notation of the time events of interest in this paper.

We denote with t_f, t_d, t_i, t_R respectively the *fault instant* (i.e., the very first time a node injects a nonzero exogenous value), *detection time*, *isolation (location) time* and *reconfiguration time*.

4.1. **Algorithm for the detection of an anomaly.** In this subsection we give a careful insight on system behavior and we exploit the intruder signature on the evolution of the monitoring nodes. The signal available to each monitoring node $i_{o_i} \in \mathcal{I}_0$ equals $y_{o_i}(t) = C_{o_i}A^t\mathbf{x}_0 + C_{o_i}A^{t-t_f}\mathbf{e}_j u(t_f) + \dots$, so a first important step is to filter out the free evolution from $y_{o_i}(t)$.

Proposition 4.1. *The autoregressive elaboration $AR(y_\ell(t))$ of the data collected from any output node is zero if data are coherent with an evolution without intruders.*

Proof: Consider the evolution of the system with no intruders, $\mathbf{x}(t) = A^t\mathbf{x}_0$. Data collected from any output node $\ell \in \mathcal{I}_0$ can be written $y_\ell = C_\ell x(t) = C_\ell A^t \mathbf{x}_0$. Then

$$\begin{aligned} AR(y_\ell(t)) &= y_\ell(t+n) + a_1 y_\ell(t+n-1) + \dots + a_{n-1} y_\ell(t+1) + a_n y_\ell(t) = \\ &= C_\ell A^{t+n}\mathbf{x}_0 + a_1 C_\ell A^{t+n-1}\mathbf{x}_0 + \dots + a_{n-1} C_\ell A^{t+1}\mathbf{x}_0 + a_n C_\ell A^t \mathbf{x}_0 = \\ &= C_\ell A^t (A^n + a_1 A^{n-1} + \dots + a_{n-1}A^1 + a_n I_n)\mathbf{x}_0 = 0 \quad \forall \mathbf{x}_0 \in \mathbb{R}^n \end{aligned}$$

where the matrix in the parentheses is the zero matrix according to the Cayley-Hamilton theorem [28].

The above result is interesting because it shows that the $AR(\cdot)$ elaboration is useful to filter out the free evolution of the system without a large amount of computation. It only requires a linear combination of the collected data.

Remark 4.1. *Notice that the $AR(\cdot)$ elaboration requires n data, so this elaboration on local data can be performed for $t \geq n$, otherwise more collaboration between the monitoring nodes is required (see Section 5.3). The latter analysis is not of interest here because the only difference of results is about the maximal detection time (see Remark 4.3) but it requires more shared data between monitoring nodes so, from the author's point of view, this choice has more drawbacks than benefits. A detailed description and comment on this aspect is reported in Section 5.3.*

If condition $AR(y_\ell(t)) = 0$ is verified at any $t \geq n$, it is *likely* that no intruder is present, but a deeper investigation is useful to have an insight on this diagnostic tool.

Some results now follow on the opposite question: *if $AR(y(\cdot)) = 0$ then is it sure that no intruder is present in the network?* And: *how often have we to test $AR(y(\cdot)) = 0$ to be sure that no intruders are present?*

The importance of this investigation is both theoretical and practical. On the one hand, this aspect received a scant attention in the distributed systems literature but it emerged and it is widely analyzed and debated in the recent literature [5, 6]. On the other hand, from a practical point of view, the importance is related to the question on how often the algorithm should be run by each agent to avoid the worst malicious behavior.

Lemma 4.1. *Let $\Sigma = (A, B, C)$ be a state space representation of a SISO dynamic system with m finite zeros. It is always possible to find a nonzero input causing a forced output coherent with a free evolution and vanishing in the first $n - 1$ time instants.*

Proof: The proof is constructive and an explicit input is built along the proof. A SISO linear system can be described by the ARMA model (10), so $AR(y(t)) = 0$ holds if and only if $MA(u(t)) = 0$. The latter equation is an homogeneous difference equation with general solution

$$u(t) = \sum_{i=1}^{\ell} \sum_{\nu=1}^{\mu_i - 1} c_{i,\nu} \binom{t}{\nu} (\rho_i)^{t-\nu} \tag{11}$$

Choose now coefficients $c_{i,\nu}$ in (11) such that $u(0) = u(1) = \ldots = u(m-1) = 0$ where $m = n - \delta_{\ell j}$. Considering the causality from input to output and the output delay $\delta_{\ell j}$ in (10), this input forces a zero output in the first $n - 1$ steps and the output signal is coherent with a free evolution because the input is a linear combination of system zeros.

Proposition 4.2. *The monitoring algorithm $AR(y_\ell(t)) = 0$, $\ell \in \mathcal{I}_0$ is coherent with the same set of initial conditions if and only if it is performed $\forall t \geq n$.*

Proof: The proof is performed via direct inspection. We first show that it is effective if it is performed at each time step, then we show a counterexample if the monitoring test $AR(y_\ell(t)) = 0$, $\ell \in \mathcal{I}_0$, is not applied in some time instants. Take a $\bar{t}$ and assume that no intruder has been present earlier, i.e., $u(\bar{t} - \delta_{\ell j}) = u(\bar{t} - \delta_{\ell j} - 1) = \ldots = 0$.

At time $\bar{t} + 1$ the condition $AR(y(\bar{t} + 1)) = 0$ implies $MA(u(\bar{t} + 1)) = 0$ and, in turn, $0 = b_1 u(\bar{t} - \delta_{\ell j} + 1) + b_2 u(\bar{t} - \delta_{\ell j}) + \ldots = b_1 u(\bar{t} - \delta_{\ell j} + 1) \Leftrightarrow u(\bar{t} - \delta_{\ell j} + 1) = 0$. Recursively, using the same arguments, if $AR(y(\tilde{t})) = 0$ is checked at every time $\tilde{t} \geq \bar{t}$ then necessarily $u(\tilde{t} - \delta_{\ell j}) = 0$ is satisfied.

Consider now the case that the test is performed at every time but at time $\bar{t}$. Choosing an input according to Lemma 4.1 starting from $\bar{t} - \delta_{\ell j}$, the evolution of the monitoring

nodes is coherent with a starting from evolution with some initial conditions $\tilde{x}_0$ until time $\bar{t}-1$ and with some others $\hat{x}_0$ from $\bar{t}+1$. In this specific evolution, $AR(y_\ell(t))=0$ holds at each time t but $\bar{t}$.

Remark 4.2 (FD Algorithm)**.** *Assume that each monitoring node has the capability to give one binary information at one time instant. Each monitoring node ℓ has to share t_{d_ℓ}, the first time the condition $AR_\ell(y(t))=0$ is not satisfied, with the others. The detection time of a non-collaborative behavior is detected at time $t_d=\min_{\ell\in\mathcal{I}_0} t_\ell$.*

Remark 4.3 (Maximum detection time)**.** *Using the above detection logic, the detection time of a misbehaving agent j is equal to $\min_{i_o\in\mathcal{I}_0} d(i_o,j)$. Maximum detection time, i.e., the greatest time interval an intruder can act within the network before an alert can be given, is $\max_{j\in\mathcal{V}}\{\min_{i_o\in\mathcal{I}_0} d(i_o,j)\}$ and it is equal to the radius of the graph $\mathcal{G}$ if no fault happens for $t<n$, or equal to n if a fault happens at $t<n$.*

Notice that the radius of a graph is the theoretical lower bound of detection time (i.e., the best achievable). Indeed, before that time no one of the output data is influenced by the exogenous signal so there is no possibility to infer the presence of a misbehaving node before.

4.2. **Localization of the misbehaving node.** As soon as an anomalous behavior of the system is detected, the major objective of a diagnostic device is the intrusion localization to exclude the misbehaving node from collaboration.

The intrusion localization module is triggered by the detection module and it is based on the set of time stamps of the detection signal from each monitoring node. Define (locally computable) signals $\varepsilon_i(t)=AR(y_i(t))$, $i\in\mathcal{I}_0$, and the associated quantities $t_{d_i}=\min_{t\geq 0}\{t \text{ s.t. } \varepsilon_i(t)\neq 0\}$.

We start from an easy basic result on the path graph.

Proposition 4.3. *Let P_n be a path graph with n nodes and monitoring set $\mathcal{I}_o=\{1,n\}$. The localization is solvable if and only if t_{d_1} and t_{d_n} are known and the estimation of the intruder location is*

$$\hat{i}=\frac{t_{d_1}-t_{d_n}+n+1}{2} \tag{12}$$

Proof: First notice from (10) that each t_{d_i} satisfies $t_{d_i}=t_f+d(i,j)+1$, where j the position of the intruder. In a path graph the two distances are $d(1,j)=j-1$ and $d(n,j)=n-j$. Therefore, we obtain

$$t_{d_1}=t_f+(j-1)+1;\quad t_{d_n}=t_f+n-j+1.$$

These are two linearly independent relations in the unknowns t_f and j, thus the solution always exists as soon as t_{d_1} and t_{d_n} are known and its computation leads to (12).

In the general case of tree graphs, things are more involved. The following lemma is useful to focus the correct localizability condition. Next, the main proposition of this section explains how to catch this condition from data.

Lemma 4.2 (Localization condition)**.** *The intrusion isolation is solvable if and only if there exist two monitoring nodes whose connection path includes the intruder node.*

Proof: Denote with t_f the fault instant, t_{d1}, t_{d2} the detection instants of two monitoring nodes i_{o1}, i_{o2} and label with j a misbehaving node. Then, $t_{d1}=t_f+d(j,i_{o1})$ and $t_{d2}=t_f+d(j,i_{o2})$. If node j does not lie along the path between i_{o1} and i_{o2} then there exists a (junction) node ℓ such that $d(j,i_{o1})=d(j,\ell)+d(\ell,i_{o1})$ and $d(j,i_{o2})=d(j,\ell)+d(\ell,i_{o2})$.

Combining the two relations:

$$t_{d1}=t_f+d(j,\ell)+d(\ell,i_{o1}),\quad t_{d2}=t_f+d(j,\ell)+d(\ell,i_{o2}).$$

It is easy to see that it is not possible to separate the unknown t_f from $d(j,\ell)$ (or, equivalently, the two relations connecting the two unknowns to data are linearly dependent) and this implies that the isolation problem cannot be solved with the knowledge of t_{d1} and t_{d2}.

On the other hand, if node j surely lies along the path between i_{o1} and i_{o2} then the arguments in the proof of (4.3) hold and the localization problem is always possible, leading to the solution (12).

Proposition 4.4. *An intruder located at a node ℓ with degree δ can be unambiguously isolated if and only if there are at least δ different t_{d_i} available from the set of monitoring nodes, each one of these coming from one connected part of $\mathcal{G} - \{\ell\}$.*

Proof: To prove that a node, say node 'κ', of degree δ cannot be localized using less than δ different t_{d_i}, $i = 1, \ldots, \delta$ consider, without loss of generality, the special case of $\delta = 3$. Define the set of nodes $\mathcal{V}_{i_o,k} = \{v \in \mathcal{V} |$ the path joining v and i_o passes through $k\}$; according to Lemma 4.2, a node k is localizable if and only if $\bigcap_{i_o \in \mathcal{I}_0} \mathcal{V}_{i_o,k} = \{k\}$. Now, if only two t_{d_i} are available, say t_{d_1} and t_{d_2}, and remembering that κ has degree $\delta = 3$, there is at least one node adjacent to node κ belonging both to $\mathcal{V}_{i_{o_1},k}$ and $\mathcal{V}_{i_{o_2},k}$. It follows that $\mathcal{V}_{i_{o_1},k} \cap \mathcal{V}_{i_{o_2},k}$ has at least two elements and so κ is not localizable. Conversely, if there is at least one t_{d_i} available from one connected part of $\mathcal{G} - \{\kappa\}$, then for any node $v \in \mathcal{V}$ it is possible to find a i_{o_i} such that $d(v, i_{o_i}) < d(\kappa, i_{o_i})$. This implies that $\bigcap_{i_o \in \mathcal{I}_0} \mathcal{V}_{i_o,k} = \{k\}$ and, in turn, that node κ is localizable.

The above proposition gives the tools to build a suitable algorithm for the localization of a misbehaving node.

4.3. **Localization algorithm.** Suppose that each monitoring node $i \in \mathcal{I}_0$ can share its local $t_{d_i} = \min_{t \geq 0}\{t \text{ s.t. } AR(y_i(t)) \neq 0\}$ with the other monitoring nodes.

1. When the first detection instant t_{d_1} happens, an anomaly of the system evolution is detected.
2. When the second detection instant t_{d_2} is available, each monitoring node computes Equation (12) applied to the path joining the two monitoring nodes that produced t_{d_1} and t_{d_2}. If the localized node is a node of degree two then the algorithm stops because the localized node is the misbehaving node (Localization condition of Proposition 4.4 is satisfied). If not, the algorithm follows on.
3. When the next detection instant, t_{d_3}, is known, then each monitoring node computes Equation (12) for all couples (t_{d_1}, t_{d_2}), (t_{d_1}, t_{d_3}) and (t_{d_2}, t_{d_3}).
4. If the localization algorithm of at least $(\delta - 1)$ couples give a localized node of degree δ then the algorithm stops because the localized node is the misbehaving node. If not the algorithm follows on waiting the next detection time, iterating points (3) and testing condition (4) of this algorithm.

Remark 4.4 (Maximum localization time)**.** *Partition the set of monitoring nodes into groups belonging to the connected subgraphs of $\mathcal{G} - \{\ell\}$, say $\mathcal{I}_0 = \mathcal{I}_{0_1} \cup \mathcal{I}_{0_2} \cup \cdots \cup \mathcal{I}_{0_\delta}$. In view of the above Proposition 4.4, the minimal localization time for a node κ of degree δ is $t_i = \max_{l \in \{1,\ldots,\delta\}} \left\{ \min_{\ell \in \mathcal{I}_{0_l}} d(\ell, k) \right\}$. A lower bound of the maximal localization time for an acyclic graph coincides with the one achievable with algorithm 4.3 and it is the diameter of the graph $\mathcal{G}$.*

5. **Fault Compensation.** As soon as the detection and localization of an anomaly is performed and a misbehaving node is found in the system, appropriate counteractions can be taken. For the systems considered in this paper, counteractions against the drift

of the collective objective function caused by the intruder (see, e.g., Equation (5)) are desirable in order to restore an unbiased group evolution.

Detection of a misbehaving node does not directly imply the exclusion of this node from collaboration. In fact when a misbehaving node is detected, the diagnostic system should check whether its behavior is intermittent or persistent. If the fault is intermittent (e.g., if an agent fails only at one instant), the monitoring nodes are able to compensate for the fault effects without a reconfiguration of the network.

In view of finding an easy yet effective compensation algorithm, we consider again the structure of data available to each monitoring node. System output for a $\ell \in \mathcal{I}_0$ is influenced by the fault according to:

$$\begin{aligned} y_\ell(t) = {} & C_\ell A^t \mathbf{x}_0 + C_\ell A^{t-t_f-1}\mathbf{e}_j u(t_f) + C_\ell A^{t-t_f-2}\mathbf{e}_j u(t_f+1) + \ldots + C_\ell A^{j-1}\mathbf{e}_j u(t-j) \\ & + C_\ell A^{j-2}\mathbf{e}_j u(t-j+1) + \ldots + C_\ell A \mathbf{e}_j u(t-2) + C_\ell \mathbf{e}_j u(t-1). \end{aligned}$$

A first, important consideration is on the zero values within the coefficients $C_\ell A^l \mathbf{e}_j$. Now, each coefficient $C_\ell A^l \mathbf{e}_j$ coincides with the forced response of the system for an impulse injected by node j at time $\bar{t}$ measured by node ℓ at time $\bar{t}+l+1$. As a consequence of the structure of the communication between agents, it is evident from direct computation of Equations (6) and (7) that an impulse at node j has an effect on node ℓ only if $l \geq d(\ell, j)$ communication rounds elapsed, so $C_\ell A^l \mathbf{e}_j = 0$ if $l < d(\ell, j)$.

This shows that each monitoring node $i_o \in \mathcal{I}_0$ can infer some knowledge on the intruder signal $\{u(\tau)\}_{|\tau \in \{t_f, \ldots, t-d(i_o,j)-1\}}$, but also that there is no way to know the other values of the intruder $\{u(\tau)\}_{|\tau \in \{t-d(i_o,j), \ldots, t\}}$.

We now look for an algorithm as simple as possible to estimate $\{u(\cdot)\}$ without estimating the whole initial condition $\mathbf{x}_0$. Still a useful tool is the output of the algorithm $AR(y(t))$ because it filters out the initial condition and it has simple connections with the intruder injections.

The collection of $\varepsilon_\ell(t) = AR(y_\ell(t))$, $\ell \in \mathcal{I}_0$ are influenced by the external input according to $\varepsilon_\ell(t_d) = AR(y_\ell(t_d)) = b_1 u(t_f)$; $\varepsilon_\ell(t_d+1) = b_1 u(t_f+1) + b_2 u(t_f)$, ... So an estimation $\hat{u}(t)$ of the external input can be inferred by any monitoring node $\ell \in \mathcal{I}_0$ using the sequence $\{\varepsilon\}_{t \geq t_f}$ once the intruder has been isolated using the recursive algorithm on local data:

$$\begin{aligned} \hat{u}(t_f) &= \frac{1}{b_1}\varepsilon_\ell(t_d) \\ \hat{u}(t_f+1) &= \frac{1}{b_1}\left[\varepsilon_\ell(t_d+1) - b_2 \hat{u}(t_f)\right] \\ &\vdots \end{aligned} \tag{13}$$

This set of equations is useful to achieve an estimation of the input at time instants t_f, t_f+2, ..., $t-d(i_o,j)-1$ using only the local information available to each monitoring node i_o.

The above estimation of the exogenous input values can be also useful for a decision on the reconfiguration of the network. Notice that, even if the system is reconfigured and the misbehaving node is excluded from collaboration at the reconfiguration time t_R, the estimation should continue until the unknowns $u(t_R-\ell)$, $u(t_R-\ell+1)$, ..., $u(t_R-1)$ are somehow estimated. A solution to this problem is given later.

We close this section with some remarks. First, the simple elaboration using only local data in Equation (13) is useful to achieve a bias compensation from any of the monitoring nodes. Details of this operation are given in the next subsection.

Second, it is also useful to derive a proper rule to decide if the misbehaving node must be excluded from the collaboration or not. By example, using relations (13), it is easy to infer if the misbehavior is intermittent or persistent. If the fault is intermittent it is possible to have an unbiased convergence using only the compensation from one monitoring node (as described later in this section). A reconfiguration of the communication graph is needed in case of a persistent fault. Indeed, a persistent correction of a persistent fault is not effective because the evolution of each state moves away from the asymptotic consensus value and the agents do not reach any asymptotic value.

Network reconfiguration schemes. Network reconfiguration is made after intrusion localization to exclude the action of the misbehaving node from system evolution. The simplest reconfiguration can be performed reassigning a zero value to the coefficients $l_{j\ell}$, $\ell \in \mathcal{N}_j$ of the neighbors of the misbehaving node in (1) (case (3) of Figure 1).

This basic kind of reconfiguration has the drawback of the disconnection of the network, condition that can be avoided if the nodes can enlarge their neighborhood with an additional amount of power consumption (case (1) of Figure 1), or if the monitoring nodes use their monitoring network (case (2) of Figure 1). From now on we assume that reconfiguration of case (1) of Figure 1 is possible. Considering for simplicity the case of misbehaving node of degree 2 as in Figure 1 case (1), the state matrix of the reconfigured system turns to:

$$A_{post_j} = \begin{bmatrix} & & & A_{j-2} & & \\ 0^T & \alpha_{j-1} & 1-\alpha_{j-1}-\beta_1 & 0 & \beta_1 & 0^T \\ 0^T & * & * & * & * & 0^T \\ 0^T & \beta_2 & 0 & 1-\alpha_{j+1}-\beta_2 & \alpha_{j+1} & 0^T \\ & & & A_{n-j+1} & & \end{bmatrix} \tag{14}$$

where A_{j-2}, A_{n-j+1} are the same rows of the matrix A in (2), β_1 and β_2 are the (new) weights between nodes $j-i$ and $j+1$ (the stars in the middle line are eventual values corresponding to the intruder behavior. Such values are related only to the eventual evolution of the intruder which no longer influences the rest of the group).

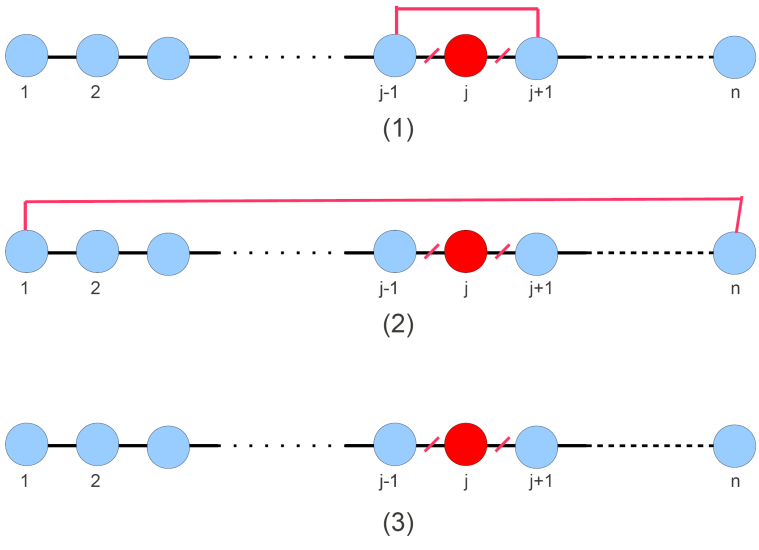

FIGURE 1. Reconfiguration schemes

5.1. **Compensation after reconfiguration.** Once a persistent misbehaving node has been localized, a reconfiguration of the communication graph is done to exclude the intruder action on the evolution of the other agents and to recover a proper unbiased system evolution. In the following, we assume that it is possible to reconfigure the communication network according to the first picture in Figure 1, but most of the following results and/or considerations hold also in general.

Our purpose of this section is to find a counteraction to the drift of the objective function caused by the intruder before its localization.

Notice that, as discussed earlier, this step is necessary to make the system evolution tend to the unbiased desired value without a restart of the system; in fact few samples of exogenous inputs or even a single impulse can compromise mission goal (5) if no counteractions are taken.

After the reconfiguration, system evolution is described by a new set of equations (Equation (2) with state matrix given by (14) for a misbehaving node localized at label j). Now, we focus on the $j-1$ unknowns that are still to be identified to make an effective fault compensation. Data available to a monitoring node at $t \geq t_R$ after reconfiguration are tied to the exogenous input according to:

$$y(t) = CA_{post}^{t-t_R} x(t_R) = CA_{post}^{t-t_R} A^{t_R} \mathbf{x_0} + CA_{post}^{t-t_R} \sum_{i=0}^{t_R-1} A^{t_R-i-1} \mathbf{e}_j u(i)$$

At a first glance, the $AR(\cdot)$ elaboration is no more effective to filter out the initial condition from the elaboration of the output signal for a $t = t_R + \upsilon$, $0 < \upsilon < n$; in fact

$$\begin{aligned} AR(y(t)) &= y(t_R+\upsilon) + a_1 y(t_R+\upsilon-1) + \cdots + a_\upsilon y(t_R) + a_{\upsilon-1} y(t_R-1) + \dots \\ &= C\left(A_{post}^{\upsilon} A^{t_R} + a_1 A_{post}^{\upsilon-1} A^{t_R} + \dots + a_{\upsilon+1} A_{post} A^{t_R} + a_\upsilon A^{t_R} + a_{\upsilon-1} A^{t_R-1} \cdots + \right. \\ &\quad \left. + a_n A^{t_R-n+\upsilon}\right) \mathbf{x_0} + C(A_{post}^{\upsilon} A^{t_R-t_f} + a_1 A_{post}^{\upsilon-1} A^{t_R-t_f} + \dots)\mathbf{e}_j u(0) + \dots \end{aligned}$$

and, since A_{post} and A do not commute, the matrix inside round brackets is no longer the empty matrix for the value of υ of interest $0 < \upsilon < n$. Notice that, if $\upsilon \geq n$, then the $AR(y)$ cancels out both the initial condition $\mathbf{x}_0$ and all the values of interest $u(\tau)$, $\tau \in [t_r - d(i_o, j), t_r]$.

A possibility to solve the identification problem is to find other coefficients $\alpha_i(\upsilon)$, $i = 1, \dots, n$ to make a different autoregressive elaboration $AR_{\alpha(\upsilon)}(y(t))$ to force the matrix $CA_{post}^{\upsilon} A^{t_r} + \alpha_1(\upsilon) CA_{post}^{\upsilon-1} A^{t_r} + \dots + \alpha_n(\upsilon) CA^{t_r-n+\upsilon}$ to be the zero matrix, but it would be computationally huge (it should be performed at each time step and on-line after the isolation has been solved because the state matrix after reconfiguration A_{post_j} depends on the located node j).

We now make a thorough analysis of the above equations with the aim of solving the above filtering procedure.

First notice that, in view of structure of the system matrix after reconfiguration (14), the following result holds. The proof is omitted because it is directly verifiable by inspection.

Lemma 5.1. *Consider matrices A in Equation (2) and A_{post_j} in (14). Considering a vector $v \in \mathbb{R}^n$ such that $(v)_\ell = 0$ for $\ell \in \mathcal{N}_j$, then*

$$v^T A = v^T A_{post_j}$$

holds.

We are now ready to deduce a simple yet useful result for simplifying Equation (15) and hence the input estimation procedure.

Proposition 5.1. *$C_{i_o} A^\upsilon = C_{i_o} A_{post_\ell}^\upsilon$ if $\upsilon \leq d(i_o, \ell) - 1$.*

Proof: The proof is conducted by direct inspection. Row vectors $C_{i_o}A^{\upsilon}$ and $C_{i_o}A^{\upsilon}_{post_\ell}$ can be computed component by component $C_{i_o}A^{\upsilon}\mathbf{e}_\kappa$ and $C_{i_o}A^{\upsilon}_{post_\ell}\mathbf{e}_\kappa$, $\kappa = 1, \ldots, n$ respectively. $C_{i_o}A^{\upsilon}\mathbf{e}_\kappa$ can be also seen as the impulse response from node κ sensed by node i_o at time $\upsilon + 1$, so $C_{i_o}A^{\upsilon}\mathbf{e}_\kappa = 0 = C_{i_o}A^{\upsilon}_{post_\ell}\mathbf{e}_\kappa$ for any υ and κ such that $\upsilon \leq d(i_o, \kappa) - 1$. On the other hand, if the condition $C_{i_o}A^{\upsilon}\mathbf{e}_\kappa \neq 0$ is satisfied, then node k necessarily satisfies $d(i_o, \kappa) < \upsilon - 1$ and, since the interval of interest is $\upsilon \leq d(i_o, \ell) - 1$, then $d(i_o, \kappa) < d(i_o, \ell) - 2$. The latter relation means that the nodes κ satisfying $C_{i_o}A^{\upsilon}\mathbf{e}_\kappa \neq 0$ within the time interval of interest are not neighbors of ℓ (since they are distant from node i_o less of two than node ℓ). Define the row vectors $v_\tau = C_{i_o}A^{\tau}$, $0 \leq \tau \leq d(i_o, \ell) - 2$. In view of the above considerations, any v_τ satisfies Lemma 5.1 conditions. Applying recursively the results of Lemma 5.1:

$$v_0 A^{\upsilon}_{post_\ell}\mathbf{e}_\kappa = v_1 A^{\upsilon-1}_{post_\ell}\mathbf{e}_\kappa = \cdots = v_\tau A^{\upsilon-\tau}_{post_\ell}\mathbf{e}_\kappa = \ldots = v_\upsilon \mathbf{e}_\kappa = v_0 A^{\upsilon}\mathbf{e}_\kappa$$

for any $\upsilon \leq d(i_o, \ell) - 1$.

So basically we proved that $C_{i_o}A^{\upsilon}\mathbf{e}_\kappa = C_{i_o}A^{\upsilon}_{post_\ell}\mathbf{e}_\kappa$ if $\upsilon \leq d(i_o, \ell) - 1$ and for any $\kappa \in \{1, \ldots, n\}$, thus we proved the statement.

Remark 5.1. *The impact of the result in Proposition 5.1 on the estimation of the exogenous input after system reconfiguration is important: it is possible to apply the easy recursive rule in (13) to make the estimation of the intruder also after system reconfiguration.*

Notice that the steps of identification of the exogenous signal require an elaboration of local information only, so the identification itself does not need any collaboration between the monitoring nodes.

5.2. **Compensation of drift on the objective function.** Once the identification of the exogenous signal is performed, the monitoring nodes can counteract the drift caused by the intruder.

First of all assume that the weights are chosen such that γ is also eigenvector of the eigenvalue in 1 of A_{post_j} (e.g., as far as the average consensus is concerned, this can be easily obtained by choosing $\beta_1 = \beta_2$).

Consider that a node, say the first monitoring node, injects an impulse of amplitude ν; system evolution shows a drift relative to the convergence value equal to:

$$\gamma^T x(t) = \gamma^T x(t_r) + \gamma^T \mathbf{e}_{i_{o_1}}\nu = \frac{\gamma^T \mathbf{x}_0 + (\gamma)_j \sum_{i<t_r} \bar{u}(i) + (\gamma)_{i_{o_1}}\nu}{\gamma^T \mathbf{1}}$$

where the last equality holds according to relation (5). As a result, after the identification step, any monitoring node (say, the first one i_{o_1}) can correct the bias introduced by the intruder on the objective function choosing $\nu = -\dfrac{(\gamma)_j}{(\gamma)_{i_{o_1}}} \sum_{i<t_r} \hat{u}(i)$, $\hat{u}(i)$ computed according to the estimation procedure (13).

5.3. **Discussion on the proposed algorithm: computational issues, conditions for the applicability, peculiarities with respect to the state of the art.** In this section, we briefly discuss about the main features of the proposed algorithm. First, we briefly summarize the steps for the correct implementation of the results in the paper with focus on the difference between on-line and off-line computation, elaboration of local data or shared data and computational complexity of each on-line elaboration.

- *Preliminary steps.* As a preliminary step, as soon as protocol (1) is chosen, coefficients of the $AR(\cdot)$ elaboration (10) are computed. It is an off-line computation and it can be performed before the system evolution starts. All the monitoring nodes

must apply the same elaboration to the sequence of local data, so it can be easily performed only once by an external computer and plugged into the memory of the agents.

- *Election of the monitoring nodes.* According to Theorem 3.2, each node of degree one is elected as a monitoring node.
- *Monitoring the network.* Each monitoring node ℓ computes the scalar sequence $\varepsilon_\ell(t) = AR(y_\ell(t))$. This elaboration is performed by each monitoring node using local information. According to results of Lemma 4.1 and Proposition 4.2, this elaboration should be performed on-line and at each time step. The computational effort of the elaboration is low (n multiplications between scalars and $n-1$ sums) and it is proportional to the number of agents (so its complexity is $\Theta(n)$).
- *Fault detection.* As soon as a monitoring node computes a nonzero value of $\varepsilon_\ell(t)$ (or, in practice, a value exceeding suitable threshold; see the example for a thorough discussion on the practical implementation of $AR(y_\ell(t)) = 0$), it broadcasts an alert to the network, thus communicating t_{d_i} to the network nodes. As soon as at least one node has given the alert, an anomalous behavior of the network is promptly detected (according to Propositions 4.1 and 4.2).
- *Fault localization.* Localization of the misbehaving node is performed as soon as the shared data t_{d_i} are available according to the algorithm 4.3 (which summarizes the results of Propositions 4.3 and 4.4). the algorithm 4.3 can be performed by any node of the network, so it is plausible that in many applications every node of the network can localize the misbehaving node.
- *Fault identification.* According to Equation (13), any monitoring node can estimate the exogenous injection of the misbehaving node. Also this operation requires computation of complexity $\Theta(n)$. The network is reconfigured if the fault is persistent. According to Proposition 5.1, the reconfiguration of the network does not change the structure of the estimation algorithm (i.e., its coefficients).
- *Fault compensation.* As far as the estimation stops, it is possible to drive the evolution to an unbiased value of consensus according to results in Section 5.1.

Conditions for the applicability of the proposed diagnostic algorithm. Here we thoroughly investigate the scope of the proposed algorithms, so we discuss in detail the conditions for the applicability of each main result.

A first condition is to deal with *acyclic graphs*. Results based on this condition are Lemma 3.1 and Theorem 3.2 which lead to the simple conceptual/logical architecture of the diagnostic system. The choice of this class of graph has been widely motivated in the introduction, basically because most studies available in the literature have considered the single node case in a highly connected network but many applications show that low connectivity networks are useful. It is worth noting that other results of the paper still hold without this hypothesis.

A second standing assumption is that agents run a *consensus protocol*. This is tightly tied to the possibility of inferring the presence of an anomaly comparing the actual evolution to a *nominal evolution*. In general, the less conservative assumption is that all nodes should agree to a known protocol, but we made this choice in view of the important applications of consensus networks in many different engineering applications.

Finally, the assumption of a single misbehaving node is posed for the ease of localization within a 1-connected network, where there is no logical redundancy so the localization problem for many misbehaving nodes requires additional information between monitors. It is worth noting that the actual assumption is that the misbehaving nodes have a rate no greater than $1/D$, D being the diameter of the Graph. Indeed, according to Remark

4.4, D is the localization time in the worst case and if different faults happen with a rate lower than $1/D$ then the algorithm can be applied recursively.

A final comment is given on the choice of using only the $AR(\cdot)$ elaboration, thus monitoring the network for $t \geq n$. According to Remark 4.1, this is just a choice of the author for the sake of simplicity of the algorithm, mainly to limit the shared information to the t_{d_i} signals. In fact, a refinement of the algorithm requires that, from $t \geq 0$, at time $\bar{t}$ each monitoring node should progressively infer the state of the $\bar{t}$-neighbor nodes and exchange this information to the other monitoring nodes. If there is a misbehaving node these data are not coherent and it is possible to infer the presence and the location of a misbehaving node within at most D time steps, D the diameter of the graph, with techniques similar to those described in [5].

Peculiarities with respect to the state of the art. We now describe the main peculiarities of the proposed algorithm with respect to the other results available in the literature. In view of the large amount of documents of this topic, we explicitly consider comparisons with two recent publications [5, 6] where the methodologies are similar to those here proposed and results can be clearly compared. Notice that the features of the proposed algorithm here described are useful to compare the performances of it with any other algorithm.

A first important peculiarity is about *response promptness* of the algorithm with respect to a fault. In [6] three detection filters are proposed. The first is a dynamical system of dimension equal to the number of agents, the second requires a number n of residual generators that each node needs to design, the third is a local detector for clusters with low interactions. Using the first and third kind of filter there is no finite time convergence of the decision filter. In the first case, the filter response has an infinite-time transient response decaying to zero, so the presence of a misbehaving node can be inferred only if it exceeds the transient response (notice that a bound on the initial conditions is necessary for this step) and moreover the authors claim that there may exist exponentially decaying inputs that may remain undetected. Analogous considerations hold for the third filter, where the interactions between clusters may alter the correct detection. In [5], the authors propose an algorithm to batch process data, so their algorithm would promptly identify an anomalous behavior in the network only if the batch is built and run near the time instant $t_f + D$, D being the diameter of the graph.

A second important peculiarity is about *low complexity* and *ease of computation* of the algorithm, which is $\Theta(n)$ for each monitoring node. The first and second filters in [6] are $\Theta(n^2)$ and $\Theta(n^3)$ respectively. The algorithm in [5] is based on n simultaneous estimation of state and misbehaving input, considering each node as a misbehaving input. The complexity of the algorithm is approximately $\Theta(n^4)$ (it depends on the method for the inversion of a $(n + \Delta) \times (n + \Delta)$ matrix, with $\Delta > 1$).

A final feature of the algorithm to consider is *recursivity*. It is a valuable property to achieve a good response promptness when algorithms have guaranteed finite-time convergence. Algorithms in [6] are recursive, the elaboration in [5] is an algorithm to batch process data.

6. **A Practical Example.**

In this section we report some simulation results on a practical example where the proposed diagnostic algorithm is applied to two consensus-based applications for a multi-agent system.

In order to evaluate the Radon pollution of an area, a team of mobile robots is designed with the mission of estimating the average value of Radon and compute the deviation of each local measure with respect to the average. To achieve this goal each agent takes a local measure and then they run the consensus algorithm (1) with unitary weight $l_{ij} = 1$.

Moreover, from time to time, the agents have to reach a common point to make a battery recharging all together. This is also performed by a consensus-based algorithm (the so called 'rendez-vous' of the robots).

The first group of figures refers to the case of an fault/intrusion within the group during the rendez-vous. The red node is the faulty one, and in the simulations a constant input is forced to the faulty node from time $t = 350$ on. It is worth noting that, if no intrusion detection module is available, the whole group is conducted outside the area of interest. Last picture shows the same evolution when the proposed algorithm runs. It is evident that the faulty node is promptly excluded from collaboration.

Finally, we apply the algorithm proposed in the paper to the distributed computation of the average of Radon. In order to test the diagnostic algorithm in a more realistic scenario, we consider the presence of uncertainties both in the state updating rule and in the measurements; we added a random variable with uniform distribution in $[-\eta, \eta]$ ($\eta = 1$ in the simulations). The idea is that bounded uncertainties can model a variety of phenomena that can concur to give slight deviations from the ideal model (6) and (7) (e.g., quantization error in elaboration and transmission, small deviations from 1 when assigning the input).

$$\mathbf{x}(t+1) = \mathbf{A}\mathbf{x}(t) + \begin{bmatrix} \bar{w}_1(t) \\ \bar{w}_2(t) \\ \vdots \\ \bar{w}_n(t) \end{bmatrix} \tag{15}$$

$$y_\ell^m(t) = y_\ell^{nom}(t) + \eta_\ell(t), \qquad \text{where} \qquad y_\ell^{nom}(t) = C_\ell x(t) \tag{16}$$

$\eta_\ell(t)$ and each $\bar{w}(t)_i$ a stochastic variable with uniform distribution in $[-\eta, \eta]$ (in the simulations we take $\eta = 1$). Here we adapt the diagnostic algorithm to this scenario by properly selecting a threshold M_ε:

$$t_{d\ell} = \min\{t : |AR(y_\ell(t)) \geq M_\varepsilon \quad M_\varepsilon = M_1 + M_2\}$$

where M_1 and M_2 account for, respectively, a bound for the diagnostic elaboration of the model uncertainty and the measurement uncertainty.

In order to find a reasonable value for M_1, consider that $\bar{w}_j(t)$ at each node contribute to the $AR(y_\ell(t))$ elaboration according to $AR(y_\ell(t)) = MA_j(w_j(t))$, $j = 1, \ldots, n$. By the superposition principle, if no misbehaving node is present, then each monitoring node computes $AR(y_\ell(t)) = \sum_{i=1}^n MA_i(w_i(t))$. Since $|AR(y_\ell(t))| = |\sum_{i=1}^n MA_i(w_i(t))| \leq \sum_{i=1}^n |MA_i(w_i(t))| \leq n\eta \sum_{i=1}^m |b_i|$, we choose $M_1 = n\eta \sum_{i=1}^m |b_i|$. As regard for the measurement uncertainty, notice that the $AR(\cdot)$ elaboration is linear, so $AR(y_\ell^m(t)) = AR(y_\ell^{nom}(t)) + AR(\eta_\ell(t))$, and since $|AR(\eta_\ell(t))| \leq n\eta \sum_{i=1}^n |a_i|$ then it is reasonable to choose $M_2 = n\eta \sum_{i=1}^n |a_i|$.

We made simulations that validate the theoretical results described along the paper. In the simulation here reported, a constant fault equal to 40 is forced on node 5 from time 40. Notice that the blue bold line represents the average of the initial conditions, so it is the asymptotic value to which every state component should converge.

Results are shown in Figure 3, where on the left the evolution with a diagnosis without compensation is shown, and finally on the right, the diagnosis is followed by the compensation from node 1. Specifically, it is clear that, if a fault turns the input of an agent to a constant value, the evolution of the whole group is compromised if no counteractions are taken. A further consequence of the presence of a misbehaving node (even if it was present only for few instants) is a drift of the asymptotic value of the group, that can be effectively be compensated by the action of the monitoring network.

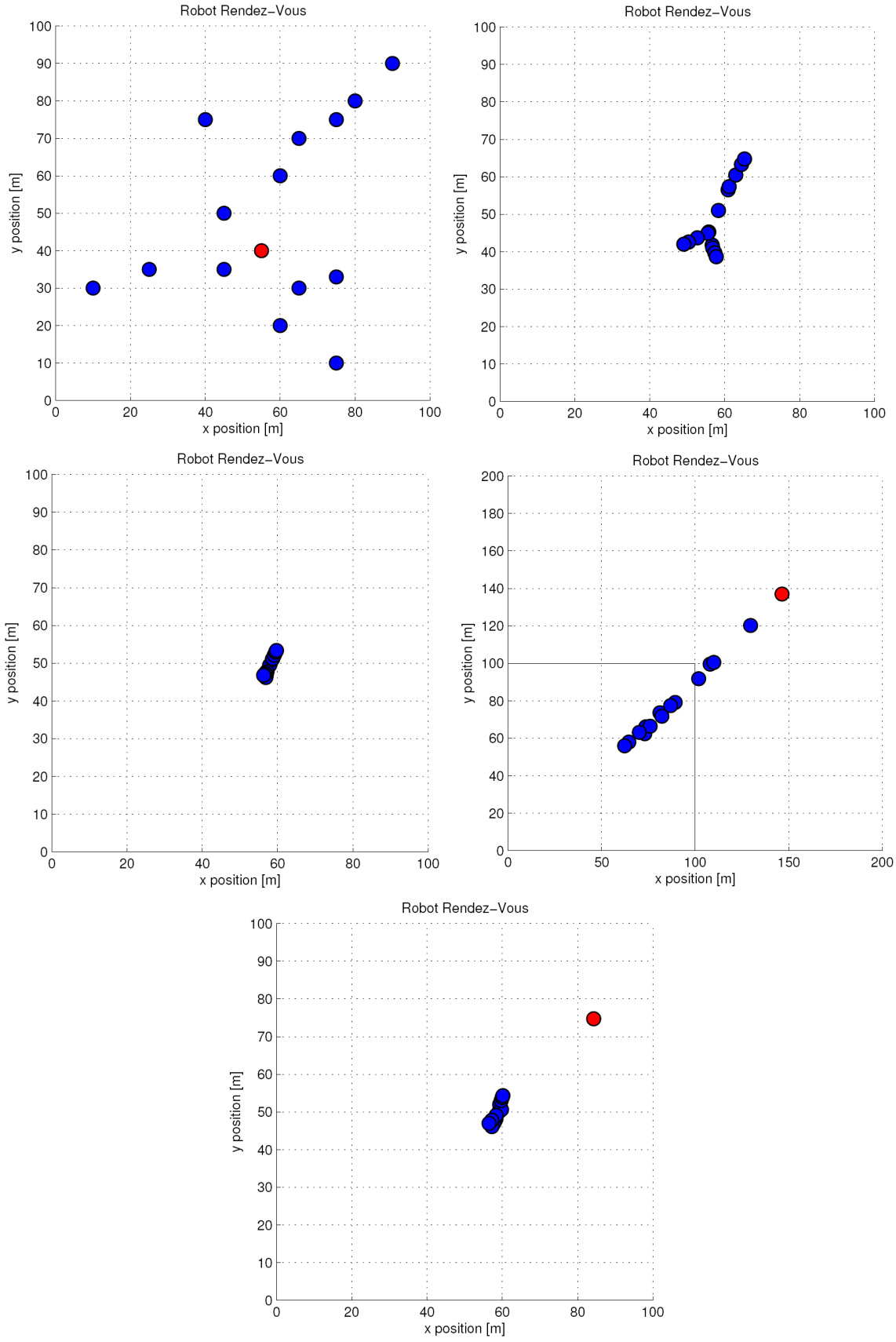

FIGURE 2. Simulation results: robot rendez-vous

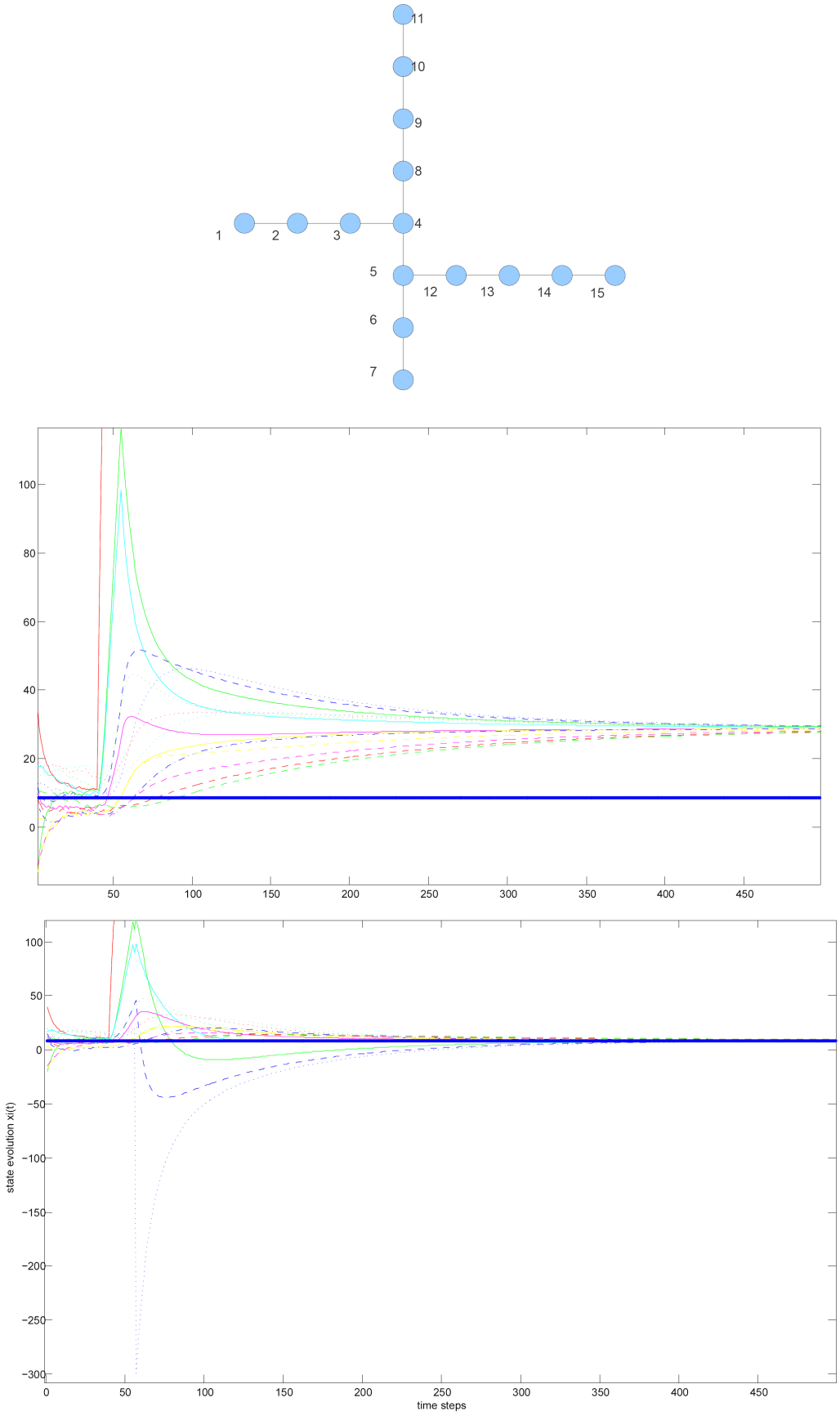

FIGURE 3. Simulation results: average consensus

7. **Conclusions.** In this paper the problem of the compensation for the action of a misbehaving node in a networked system with acyclic communication graph subject to the presence of an eventual fault/intrusion is considered. A thorough analysis of a diagnostic tool is investigated and a simple multinode strategy is proposed. The solution has the architecture of a collaborative multinode with precise and easy instructions for each node of the monitoring network. Implementing issues are investigated, considering the adaptation of the diagnostic algorithm in the presence of bounded uncertainties.

REFERENCES

[1] F. Bullo, J. Cortés and S. Martínez, *Distributed Control of Robotic Networks, Applied Mathematics Series*, Princeton University Press, 2009.

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

[3] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation*, Englewood Cliffs, Prentice-Hall, NJ, 1989.

[4] L. Lamport, R. Shostak and M. Pease, The byzantine generals problem, *ACM Transactions on Programming Languages and Systems*, vol.4, no.3, pp.382-401, 1982.

[5] S. Sundaram and C. N. Hadjicostis, Distributed function calculation via linear iterative strategies in the presence of malicious agents, *IEEE Transactions on Automatic Control*, vol.56, no.7, pp.1495-1508, 2011.

[6] F. Pasqualetti, A. Bicchi and F. Bullo, Consensus computation in unreliable networks: A system theoretic approach, *IEEE Transactions on Automatic Control*, vol.57, no.1, pp.90-105, 2012.

[7] C. Zhang, X. Bai, J. Teng, D. Xuan and W. Jia, Constructing low-connectivity and full-coverage three dimensional sensor networks, *IEEE Journal on Selected Areas in Communications*, vol.48, no.7, pp.984-993, 2010.

[8] S. Liang, Y. Tang and Q. Zhu, Passive wake-up scheme for wireless sensor networks, *ICIC Express Letters*, vol.2, no.2, pp.149-154, 2010.

[9] J. Deng, K. Deng, Y. Li and T. Sun, The robust synchronization problem of complex dynamical networks, *ICIC Express Letters, Part B: Applications*, vol.2, no.5, pp.1027-1032, 2011.

[10] M. Qian and J. Cao, Robust fault-tolerant control for a class of spacecraft rendezvous system: Actuator fault case, *ICIC Express Letters, Part B: Applications*, vol.1, no.2, pp.255-261, 2010.

[11] W. Ren, R. W. Beard and E. M. Atkins, A survey of consensus problems in multi-agent coordination, *ACC*, Los Angeles, CA, USA, 2006.

[12] A. K. Somani, System level diagnosis: A review, *Technical Report*, 1997.

[13] Z. Fan, An active management framework for automatic fault detection and elimination in distributed systems, *ICIC Express Letters, Part B: Applications*, vol.2, no.1, pp.31-37, 2011.

[14] Z. Mao and B. Jiang, Fault identification and fault-tolerant control for a class of networked control systems, *International Journal of Innovative Computing, Information and Control*, vol.3, no.5, pp.1121-1130, 2007.

[15] P. Brutch and C. Ko , Challenges in intrusion detection for wireless ad-hoc networks, *Proc. of 2003 Symposium on Applications and the Internet Workshops*, pp.368-373, 2003.

[16] H. Yan, H. Luo, F. Ye, S. Lu and L. Zhang, Security in mobile ad hoc networks: Challenges and solutions, *Wireless Communications, IEEE*, pp.38-47, 2004.

[17] Y. Zhang, W. Lee and Y. A. Huang, Intrusion detection techniques for mobile wireless networks, *Wireless Networks*, vol.9, pp.545-556, 2003.

[18] W. Lin, L. Xiang, D. Pao and B. Liu, Collaborative distributed intrusion detection system, *The 2nd International Conference on Future Generation Communication and Networking*, pp.172-177, 2008.

[19] C. Zhang, X. Bai, J. Teng, D. Xuan and W. Jia, Decentralized multi-dimensional alert correlation for collaborative intrusion detection, *Journal of Network and Computer Applications*, vol.32, pp.1106-1123, 2009.

[20] J. Partan, J. Kurose and B. N. Levine, A survey of practical issues in underwater networks, *WUWNet*, Los Angeles, CA, USA, 2006.

[21] Y. Gong, X. Wang, R. He, F. Pang and C. Gao, The design and application of wireless ad hoc maritime communications system, *ICIC Express Letters*, vol.5, no.2, pp.461-466, 2011.

[22] S. Raghavan, Low-connectivity network design on series-parallel graphs, *Networks, Wiley Periodicals*, vol.43, no.3, pp.163-176, 2004.

[23] Z. Ji, H. Lin, T. H. Lee and Q. Ling, Multi-agent controllability with tree topology, *Proc. of 2010 American Control Conference*, Baltimore, MD, USA, 2010.
[24] G. Parlangeli, Further considerations on the intrusion detection in an average consensus networked system: Multinode design for acyclic graphs, *Proc. of the 18th Mediterranean Conference on Contr. Autom. (MED)*, Marrakech, Morocco, 2010.
[25] C. Godsil and G. Royle, Algebraic graph theory, *Graduate Texts in Mathematics*, vol.207, 2001.
[26] F. Harary, *Graph Theory*, Reading, Addison-Wesley, MA, 1994.
[27] J. Tokarzewski, *Finite Zeros in Discrete Time Control Systems*, Springer, 2006.
[28] F. Gantmacher, *The Theory of Matrices*, Chelsea, New York, 1960.