Decentralized Fault Diagnosis for Sensor Networks

Mauro Franceschelli, Alessandro Giua, Carla Seatzu

Abstract

In this paper we study the problem of fault diagnosis for sensor networks. We examine faults that involve an anomalous behavior of the sensor and investigate their diagnosis only through the local interaction between faulty nodes and healthy ones. We provide heuristics to actively diagnose faults and recover the nominal behavior.

Published as:
M. Franceschelli, A. Giua, C. Seatzu " Decentralized Fault Diagnosis for Sensor Networks," *5th IEEE Conference on Automation Science and Engineering*, (Bangalore, India), Aug. 2009.

M. Franceschelli, A. Giua and C. Seatzu are with the Dept. of Electrical and Electronic Engineering, University of Cagliari, Piazza D'Armi, 09123 Cagliari, Italy. Email: mauro.franceschelli@diee.unica.it, {giua,seatzu}@diee.unica.it.

I. Introduction

In the last decade a great deal of attention has been drawn toward distributed sensor networks from researchers around the world. The technological feasibility of small, cheap and wireless sensing devices allows to envision applications as large scale sensor networks for weather measurement and forecasting, a more precise and detailed pollution monitoring in industrial districts and dumps, distributed measurements for electromagnetic pollution, space applications that require high robustness to sensor failures and so on. Apart from technological advances that allow the production of smarter devices that consume little power, there is a fundamental need for efficient distributed algorithms for sensor fusion and estimation to gain the most from such devices. One reason why new estimation algorithms are needed is that such networks are intended to be *large-scale*, i.e., the number of connected devices is expected to be large so as to cover large surface areas like a city or a park or a nation. Such networks need scalable algorithms, i.e., algorithms whose computational complexity growths at most linearly with respect to the number of network nodes, and most of all, decentralized algorithms able to solve problems addressing the topological communication constraints of these networks and able to distribute evenly the computational workload to exploit parallelism. One significant example of these algorithms is the consensus algorithm [9]. Such algorithm has as objective to retrieve the average of the measurements from a sensor network by performing iteratively local averages with neighboring nodes.

A great effort has been put by researchers in control systems to study and improve the convergence rate of the consensus algorithm in his many forms [10]. Another line of research has been to study the effects of noise [15], quantization errors [16], switching topologies and time delays [13]. Recently many researchers have studied the problem of fault detection on sensor networks from different points of view. In [18] is studied the problem of fault detection regarding network congestion, hidden terminals or asymmetric links. In [20] a method for the fault diagnosis of a pulp mill process is developed. This is one of the largest processes in the fault diagnosis area and it was proposed as a new benchmark process to evaluate the fault diagnosis method. In [21] it is developed a framework for distributed model-based fault diagnosis using sensor networks and consensus-based filter. The originality of this work is to allow the sensors to perform sensible computations in a distributed fashion implementing a distributed Kalman Filter and distributed data fusion algorithms.

In this paper we focus on a relevant weakness of the consensus algorithm used to solve the distributed average problem for data fusion. Namely even if a network is composed by millions of nodes implementing the consensus algorithm, if just one of them does not implement it correctly due to hardware or software bugs, it prevents the network from converging properly to the desired estimation. In this paper, following a previous work by the authors about fault recovery in sensor networks [14], we add robustness to the consensus algorithm by providing a method of active fault diagnosis and identification to be paired with an already published fault recovery algorithm. The main contribution of this work is:

1) A classification of faults in a sensor network that disrupt the consensus dynamics.
2) A heuristic to classify suspected behavior of the nodes when a fault is likely.

3) The application of a method developed in [14] to actively probe the nodes suspected to be faulty.
4) An heuristic to identify faulty nodes once they have been probed.
5) The application of a method developed in [14] to recover the network convergence property after a fault has been detected.

II. Problem Statement

We will consider a sensor network, whose nominal state evolution is governed by a discrete time consensus equation that can be written quite generally as

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

where P is a stochastic, indecomposable, aperiodic matrix, as discussed in [9]. Moreover, $x \in \mathbb{R}^n$ is an aggregated state vector, with each component x_i representing a scalar state associated with agent $i = 1, \dots, n$.

We model the network as an undirected graph $\mathcal{G} = V \times E$, with V being a set of vertices $V = \{1, \dots, n\}$ that represent the agents, and where the edge set $E \subseteq V \times V$ encodes the network topology in that $(i, j) \in E$ if and only if agents i and j can share information. The graph can be encoded through its adjacency matrix A, i.e., a $n \times n$ matrix such that $a_{i,j} = 1$ if and only if $(i, j) \in E$ and is 0 elsewhere [3].

Let $\mathcal{N}_i \subset V$ be the set of vertices adjacent to vertex i, and let $|\mathcal{N}_i|$ denote its cardinality. We can then define the degree matrix Δ as the diagonal matrix whose diagonal entries are $\Delta_{i,i} = |\mathcal{N}_i|$. Using these matrices, a standard, discrete time model of consensus networks is the one defined by $x(k+1) = (I - \varepsilon\mathcal{L})x(k)$, where I is the identity matrix, $\mathcal{L} = \Delta - A$ is the graph Laplacian of the graph $\mathcal{G}$, and $\varepsilon > 0$ the sampling time. Under this dynamics, the matrix P in Equation (1) becomes

$$P = I - \varepsilon\mathcal{L}. \tag{2}$$

We consider two different kinds of faults.

1) *Stuck at* : This fault is usually due to faults in the electronics of the sensor: it consists in a sensor that is "stuck" at an arbitrary value. This makes the nodes that correctly follow the prescribed communication protocol drift away toward the faulty node value.
2) *Divergence fault* : This fault is due to software or hardware failure: it consists in a sensor constantly increasing (or decreasing) its state value indefinitely (or until a saturation occurs). This makes the nodes that correctly follow the communication protocol drift away in the direction of the faulty node value.

III. The motion probe

In this section we briefly recall the "Motion Probe", namely a way to perturb a network performing consensus while not disrupting its averaging properties. Such result was firstly introduced in [14].

In order to allow for the sensors to exert a control action different from the consensus-based, we assume that the network behavior can be described by:

$$x(k+1) = Px(k) + u(k), \qquad x(0) = x_0. \tag{3}$$

Here $u \in \mathbb{R}^n$ is a vector of inputs whose i^{th} component u_i is the scalar input exerted by node i.

Theorem 1: [14] *Given the network dynamics in Equation* (3). *If*

$$\sum_{k=0}^{t-1} u(k) = \mathbf{0},$$

then $\mathbf{1}^T x(t) = \mathbf{1}^T x(0)\ \forall t > 0$.

The above result implies that each node may locally apply any input during the transient response of the system to actively identify faulty behavior without disrupting the convergence properties if the integral of such input is null. We call any perturbation of the network state satisfying Theorem 1 "Motion Probe".

We point out that since different nodes may apply the motion probe at the same time, to avoid rare situations in which motion probes from different nodes nullify their contribution on a given node they can be performed with a random amplitude.

IV. Active diagnosis and recovery of decentralized sensor networks

Our approach for diagnosis and recovery can be summarized into three steps that the single nodes can apply independently and asynchronously:

1) *Detection of suspicious behavior* : This step is performed by each node toward each of its neighbors. The detection of suspicious behavior is carried out by observing the neighbors' states and checking through a set of rules the presence of anomalous behavior.
2) *Fault identification* : This step is performed by the nodes who identified a suspicious behavior in one of their neighbors. It consists in applying an appropriate input to the network and check the reaction of the suspected node to identify an eventual fault.
3) *Fault recovery* : This step is performed by each neighbor of a faulty node once its faulty behavior has been detected. It allows the sensor network to disconnect the faulty node and recover the average of the initial states as if the faulty node had never influenced the network dynamics.

A. Detection of suspicious behavior

In our approach each node constantly observes the neighbors states checking for suspicious behavior. In the following we identify some behaviors associated with their respective fault and we will refer to them as "suspicious" throughout the rest of the paper.

1) *Stuck at* : This fault is characterized by a node that doesn't update anymore its state but remains visible to its neighbors. To identify such behavior is sufficient to check whether the state of a particular node is not equal to the neighbors' state and it is not changing. In the general case the state of a healthy node may not change only if it already corresponds

to the average of the initial states and the rest of the network is in a very unlikely trajectory that keeps the input of such node always equal to zero. A rule of the thumb to detect such behavior is simply to check whether the neighbor state is not changing for a sufficiently long time despite the difference between the node state and the suspected node state is different from zero and greater than an appropriate ε small enough such that this kind of behavior can be detected before all the nodes converge to the same value.

2) *Multiple stuck at faults* : In this case as discussed in [5], the states of the nodes converge toward a value in the convex hull spanned by the nodes stuck at a certain value. This kind of fault is easily detected by observing that the nodes do not reach the same state after a sufficiently long time (where for sufficiently long we mean $4-5$ times the slowest time constant given by the second largest eigenvalue that corresponds to the algebraic connectivity of the graph representing the network).
3) *Divergence fault* : This fault is characterized by an indefinite constant increment (or decrement) of the node's state. This kind of fault can be due for instance to software or hardware bugs. This fault prevents the network to converge toward a common value. As such it can be identified by detecting the sustained increments (or decrements) of the node through model based identification techniques.

B. Fault detection

After a node has been tagged "suspected" its neighbors may perform a test to verify its behavior. Such test consists in applying an input satisfying the condition of Theorem 1 and observing its reaction. A suitable input is a single square wave perturbation. If the suspect fault is the "stuck at" or the "divergence fault" then any reaction to this input means that the node is healthy, while no reaction means a node failure.

C. Fault recovery

If we assume that we have been able to locate a faulty node (e.g. using the motion probe discussed in the previous section), what we would like to do is to isolate that node from the network. Moreover, we would like to not only cancel out that node's effect on the system after recovery, but also to nullify the nodes total effect, from time $t = 0$ onwards. The reason why this might be useful can for instance be seen in a networked robot system where we do not want to let the faulty node drag the team away from the desired rendezvous point. Similarly, in a sensor network, we typically need to eliminate the contribution from a faulty sensor.

We assume that the nodes are ordered in such a way that the first $n-m$ nodes are non-faulty, and the remaining m nodes are faulty. In fact, we let the system dynamics be given by

$$x(k+1) = \hat{P}x(k) + u(k), \qquad x(0) = x_0, \tag{4}$$

where $\hat{P}$ can be expressed as follows. Let W_f be the $n \times n$ matrix whose entries are zeros except the bottom right $n \times m$ block which is the identity matrix:

$$W_f = \left[\begin{array}{c|c} \mathbf{0}_{(n-m)\times(n-m)} & \mathbf{0}_{(n-m)\times m} \\ \hline \mathbf{0}_{m\times(n-m)} & I_{m\times m} \end{array}\right].$$

Using this notation $\hat{P}$ becomes $\hat{P} = I - \varepsilon(\mathcal{L} - W_f\mathcal{L})$, where $\mathcal{L}$ is the graph Laplacian.

It is straightforward to show that $\hat{P}$ in Equation (4) can be written as the following, partitioned block matrix:

$$\hat{P} = \left[\begin{array}{c|c} P_g & D_f \\ \hline \mathbf{0}_{m\times(n-m)} & I_{m\times m} \end{array}\right]. \tag{5}$$

Here D_f is a $(n-m)\times m$ matrix whose elements in the i^{th} row are non-zero only corresponding to faulty nodes neighbors of node i.

In the following we denote u_g the inputs to the good nodes and u_f the inputs of the faulty ones. Since $\hat{P}$ is upper block triangular, we can now write the dynamics of the non-faulty nodes (denoted by x_g) and view the position of the faulty nodes (x_f) as inputs. Moreover, we recall that u_f will not affect x_g directly, and we get the following partitioned system:

$$\begin{aligned} x_g(k+1) &= P_g x_g(k) + D_f x_f(k) + B_g u_g(k) \\ x_f(k+1) &= x_f(k) + B_f u_f(k). \end{aligned} \tag{6}$$

Letting $F = \{i \mid \text{ node } i \text{ is faulty}\}$ allows us to separate the contributions to the non-faulty node's evolution. The local interaction rule implemented by the good agents can then be written as:

$$\begin{aligned} x_{g,i}(k+1) = \;& x_{g,i}(k) - \varepsilon \Big(\textstyle\sum_{j\in\mathcal{N}_i/F}(x_{g,i} - x_{g,j}) \\ & + \textstyle\sum_{j\in\mathcal{N}_i\cap F}(x_{g,i} - x_{f,j}) \Big) + u_i(k). \end{aligned}$$

From the previous description we can extract the component due to the faulty agents.

The following theorem provides a tool for network recovery after some nodes have been disconnected from the network for some reason and it is of interest to nullify their contribution to the final state of the network.

Note that in the following we denote H as a $(n-m)\times(n-m)$ diagonal matrix whose i^{th} element corresponds to the number of faulty nodes in the neighborhood of node i multiplied by ε:

$$H = \varepsilon \text{ diag}\left(\; |\mathcal{N}_1 \cap F| \quad \cdots \quad |\mathcal{N}_{n-m} \cap F| \; \right).$$

***Theorem** 2:* [14] *Let the network of nodes be described by Equation* (6)*. If the following conditions hold:*

1) *At time t_0 all m faulty nodes have been detected.*
2) *The neighbors of faulty nodes apply a control input $u_{rec}(\cdot)$ such that at time t*

$$\sum_{k=t_0}^{t} u_{rec}(k) = -\sum_{k=0}^{t_0-1}(D_f x_f(k) - H x_g(k)).$$

Then, at time t a complete recovery of the weighted average of the initial states of the non-faulty nodes has been performed, i.e.,

$$\boldsymbol{1}^T x_g(0) = \boldsymbol{1}^T x_g(t).$$

If, furthermore, the induced subgraph of $\mathcal{G}$ containing all the non-faulty nodes is connected after time t, then

$$\lim_{k\to\infty} x_g(t+k) = \frac{\mathbf{1}^T x_g(0)}{n-m} \cdot \mathbf{1}.$$

We point out the following main features of the proposed recovery procedure:

1) Only the nodes that are neighbors to a faulty node need to apply the recovery input.
2) The information needed by the generic node to recover after detection is

$$N_{ij}(k) = e_i^T (D_f e_j e_j^T x_f(k) - H x_g(k))$$

   that essentially consists of the summation of the difference between its state and the faulty neighbor's (i.e., their distance) starting from the initial instant of time, i.e., *a single variable for each neighboring node*.

   We point out that since the agents do not know that a neighbor is faulty until it is identified, they are required to keep track of one single variable for each of their neighbors from the beginning of the consensus algorithm to be able to eventually disconnect the faulty link.
3) Any node that has detected a faulty node in its neighborhood may apply the recovery at any time. There is no need that all the faulty nodes are detected at the same time. The recovery procedure may then be applied in an asynchronous way.

V. Algorithm description

We now present the actual algorithm run by each node to perform identification and fault recovery. The generic node i runs the following algorithm with parameters.

- Δ: a sliding window horizon. This value should be comprised between the slowest and fastest time constant in the system.
- ε: a threshold to determine if a node value has remained constant; it will be used to identify a "stuck at" node.
- γ: a threshold to determine if agent j is suspected to be affected by a "divergence fault".
- T_{max}: roughly an estimation of the consensus convergence time; a good choice is $4-5$ times the slowest time constant given by the graph algebraic connectivity.

***Algorithm** 1 (Diagnostic Algorithm):*

1) Let $t = 0$;
2) While $t \leq T_{max}$, do
3) For each neighbor $j \in \mathcal{N}_i$:
   If $|x_j(t+k) - x_j(t)| < \varepsilon,\ \forall k = 0, \ldots, \Delta$, or
   if $|x_j(t+1+k) - x_j(t+k)| > \gamma > 0, \forall k = 0, \ldots, \Delta$,
   label that node as *suspected*.
4) If at least one neighbor j is labeled as *suspected*, execute a *motion probe* and observe its reaction.
5) If during the execution of the motion probe, node j does not change behavior, namely if it remains stuck at its value within a threshold ε or it continues to diverge within a certain threshold γ, label it as *faulty*.

6) For each neighbor j labeled as faulty, disconnect it and execute the recovery for node j.
7) end while.
8) Stop.

VI. Numerical simulations

In this section simulations are provided to show the effectiveness of the heuristics. We take the sensor network in Figure 1 as case study for the following examples. The network consists in 9 sensors labeled $S_i : i = 1, \dots, 9$, that at time $t_0 = 0$ perform a measurement and then fuse such information by performing iterative averaging based on the Laplacian of the network graph. In the following the convergence time will be shown as number of iterations. The initial state of the network has been taken randomly with a uniform distribution between 0 and 10:

$$x(t_0) = \begin{array}{l} [0.7818; 4.4268; 1.0665; 9.6190; 0.0463 \\ 0.0463; 7.7491; 8.1730; 8.6869; 0.8444] \end{array}$$

A. Stuck at fault

In Figure 2 is shown the evolution of the state of the sensor network. At time $T_1 = 50$ the sensor node S_1 experience a fault and remains stuck at a value around 2.4. As a consequence if the fault recovery algorithm proposed in this paper is not applied we may see that all the nodes wrongly converge toward the state of the faulty node disrupting the information retrieved from the measurements.

In Figure 3 is shown the evolution of the same network when the proposed algorithm is implemented. At time $T_1 = 50$ the sensor node S_1 breaks again. The neighbors of sensor S_1, namely sensor S_2 and S_3, after some time (the amount of such time is a design parameter that has to be estimated as function of the worst case convergence time) detect the suspicious behavior of node S_1 and perform a Motion Probe. The Motion Probe performed has been chosen to be a single period lasting 20 iterations of a square wave with amplitude equal to 0.1. Both sensors after performing the Motion Probe detect the faulty behavior of node S_1 that is not reacting. Consequently they disconnect it from the network and apply the recovery algorithm nullifying its entire contribution to the network. From this point on the network evolves as a standard consensus network and converges to the average of the initial state of the healthy nodes.

B. Multiple Stuck at faults

We now study the case of multiple stuck at faults and show how the recovery algorithm can be applied asynchronously with respect to each of its phases.

In Figure 4 is shown the evolution of the sensor network in Figure 1 for the same initial conditions as for previous simulations. At time $T_1 = 50$ sensor S_1 stops updating its states. At time $T_2 = 150$ also sensor S_7 stops updating. At this point the sensor network does not converge anymore to a common value but each sensor state converges toward a point in the convex hull spanned by the faulty nodes values. In this condition any attempt to retrieve information about the original average of the measurements would fail since each node presents a different estimate.

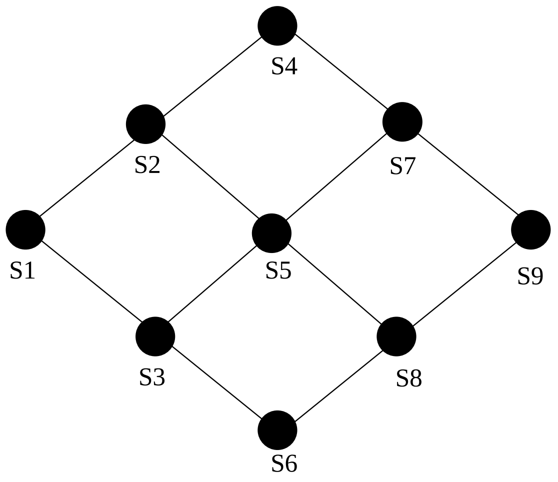

Fig. 1. The sensor network topology used for the simulations.

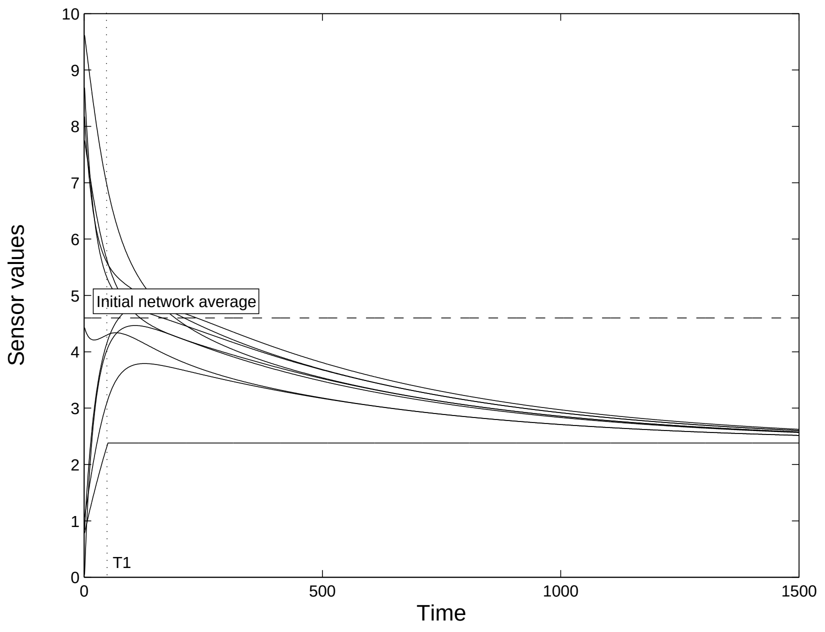

Fig. 2. Evolution of a sensor network with a single "stuck at" fault.

In Figure 5 is shown the evolution of the same network with two faulty nodes implementing the proposed algorithm for recovery. At time $T_1 = 50$ sensor S_1 is stuck; at time $T_2 = 150$ sensor S_7 is stuck as well. At time T_3 sensors S_2 and S_3 suspect of the behavior of node S_1 and perform a Motion probe. At time T_4 sensors S_2 and S_3 detect the faulty behavior of node S_1, disconnect it from the network and apply the recovery input to nullify its contribution. At time T_5 sensors S_4 and S_5 (neighbors of S_7) detect the suspicious behavior of the second faulty nodes,

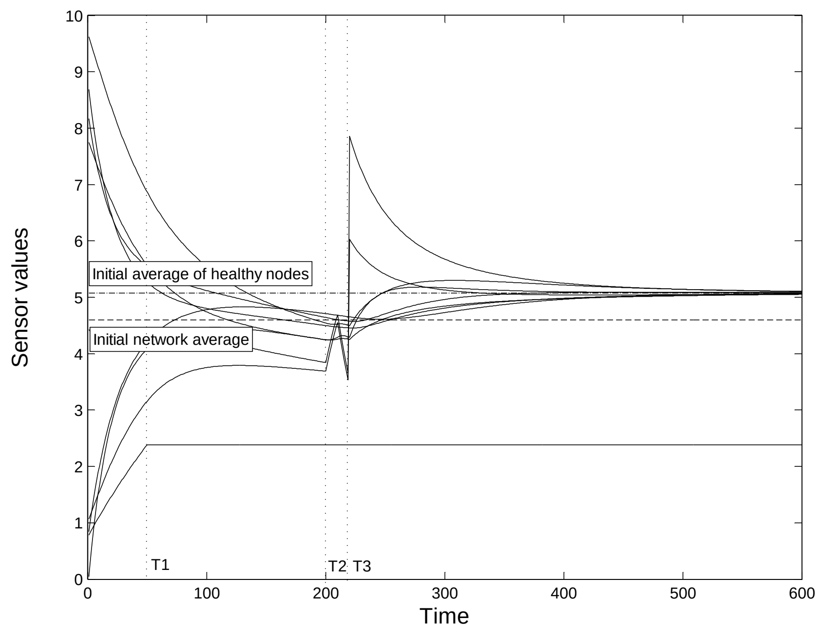

Fig. 3. Evolution of a sensor network with a single "stuck at" fault when the fault recovery procedure is implemented.

S_7, while sensor S_9 for some reason does not recognize such behavior. At time T_6 sensors S_4 and S_5 complete the motion probe, detect no reaction and disconnect form sensor S_7 nullifying its contribution. After some time also node S_9 detects the suspicious behavior of node S_7 and at time T_7 it disconnects it and applies the recovery. At this point the network of the remaining nodes, that despite the faulty nodes remained connected, converges toward the initial average of the healthy nodes' states. This simulation gives an example of how the phases of fault identification and recovery can be applied asynchronously by the neighbors of the faulty nodes.

C. Divergence faults

In this last set of simulations we show how the proposed method for fault identification and recovery can be applied also to different or more general kinds of fault.

In Figure 6 is shown the sensor network in Figure 1 performing consensus. Here at time $t = T_1$ node S_1 starts to disobey to the consensus protocol and starts to increase (in such a case, linearly) indefinitely its value due to a software or hardware bug. The remaining nodes in the network unaware of the fault start following its path and the original measurements are lost.

In Figure 7 is shown the same network with the same initial conditions as the previous examples with a fault at sensor S_1 at $t = T_1$ when the fault recovery and identification procedure is implemented. This time the neighbors of sensor S_1 detect its suspicious behavior thanks to a fault model embedded in their memory. As soon as they detect a neighbor increasing or decreasing at a certain rate they execute a motion probe and wait for the node's reaction. Since node S_1 does not change its behavior as function of the inputs given by nodes 2 and 3, it is correctly identified as faulty and disconnected from the network.

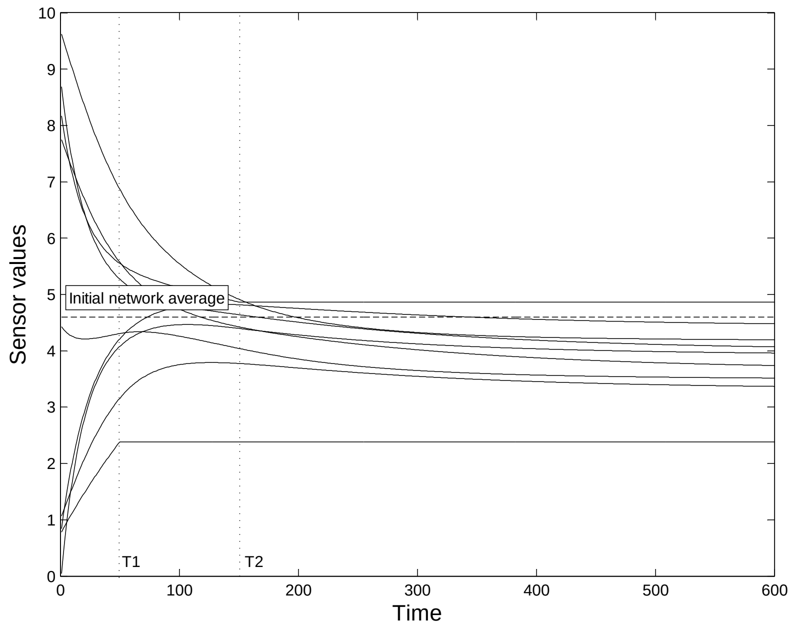

Fig. 4. Evolution of a sensor network with multiple "stuck at" faults.

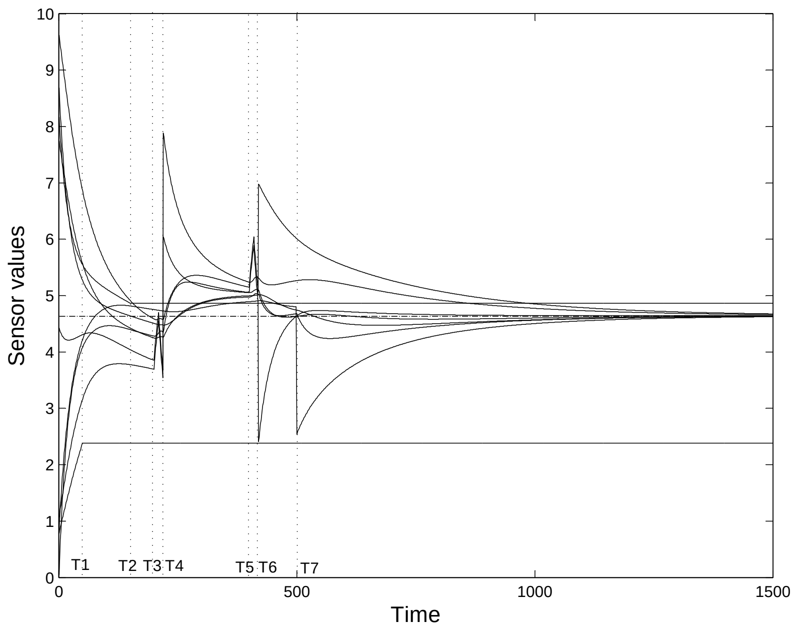

Fig. 5. Evolution of a sensor network with multiple "stuck at" faults when the fault recovery procedure is implemented.

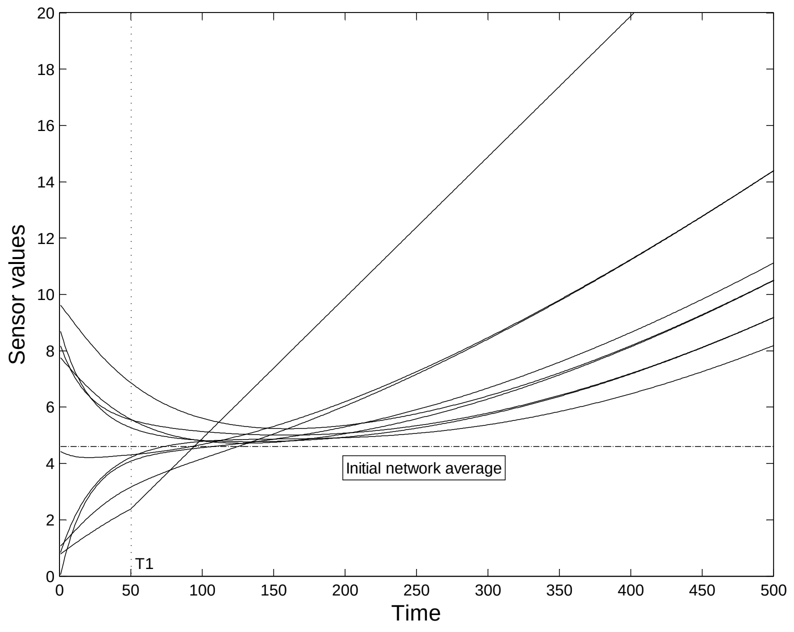

Fig. 6. Evolution of a sensor network with single "ramp" faults.

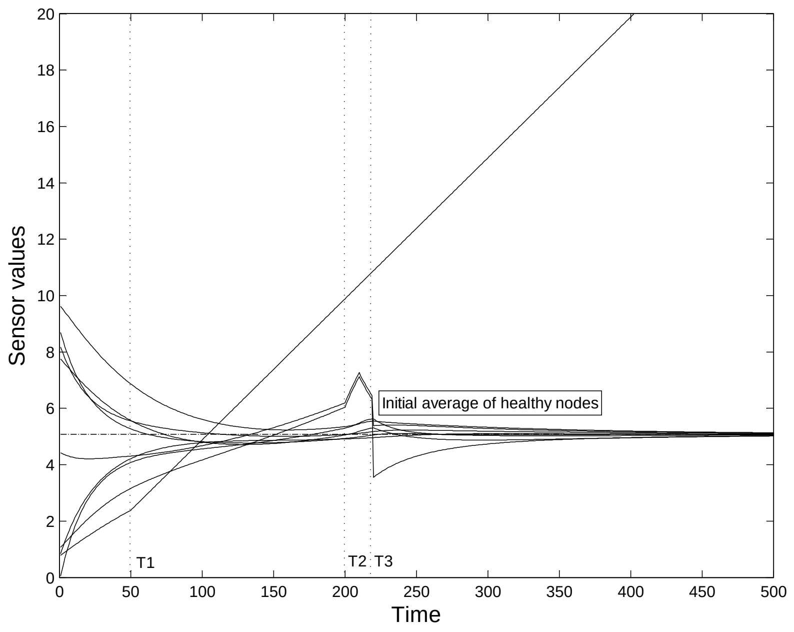

Fig. 7. Evolution of a sensor network with single "ramp" fault when the fault recovery procedure is implemented.

Remark 1: Since the algorithm is decentralized, the overhead required is proportional to the average degree of the network. If the average degree of the network is constant as the number of nodes increases, the trade-off between overhead/performance improves. Furthermore, the presence of only one faulty node disrupts the network convergence properties with any number of nodes. It becomes clear that as the number of nodes increases, the probability of node failure increases and the fault diagnosis and recovery algorithm becomes essential.

VII. Conclusion

In this paper we have provided a set of tools for decentralized fault diagnosis and recovery for sensor networks. We provided a set of heuristics for fault identification based on the faults' models. Simulation results are provided to show the effectiveness of the heuristics. The study of more general kinds of faults will be object of future research in this topic.

References

[1] A. Bensoussan and J.L. Menaldi. Difference equations on weighted graphs. *Journal of Convex Analysis* (Special issue in honor of Claude Lemarechal) 12(1), 13-44, (2005)

[2] J. Cortes, S. Martinez, and F. Bullo. Robust rendezvous for mobile autonomous agents via proximity graphs in d dimensions. *IEEE Trans. Robot. Automat.*, 51(8): 1289-1298, 2006.

[3] C. Godsil and G. Royle. *Algebraic graph theory*. Springer, 2001.

[4] G. Ferrari-Trecate, A. Buffa, and M. Gati. Analysis of coordination in multi-agent systems through partial difference equations. Part I: The Laplacian control. *16th IFAC World Congress on Automatic Control*, 2005.

[5] G. Ferrari-Trecate, M. Egerstedt, A. Buffa, and M. Ji. Laplacian Sheep: A Hybrid, Stop-Go Policy for Leader-Based Containment Control. *Hybrid Systems: Computation and Control*, Springer-Verlag, pp. 212-226, Santa Barbara, CA, March 2006.

[6] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Trans. Automat. Contr.*, 48(6):988-1001, June 2003.

[7] Z. Lin, M. Broucke, and B. Francis. Local control strategies for groups of mobile autonomous agents. *IEEE Trans. Automat. Contr.*, 49(4):622-629, 2004.

[8] R. Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Trans. Automat. Contr.*, 51(3):401-420, March 2006.

[9] R. Olfati-Saber. Consensus and cooperation in networked multi-agent systems. *IEEE Proceedings.*, 95(1):215, Jannuary 2007.

[10] W. Ren, R.W. Beard, and E. Atkins. A Survey of Consensus Problems in Multi-agent Coordination. *American Control Conference*, Portland, OR, 2005.

[11] K. Sugihara and I. Suzuki. Distributed motion coordination of multiple robots. In Proceedings of *IEEE Int. Symp. on Intelligent Control*, pages 138-143, 1990.

[12] H. Tanner, A. Jadbabaie, and G.J. Pappas. Stable flocking of mobile agents, part II : Dynamic topology. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 2016-2021, 2003.

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

[14] M. Franceschelli, M. Egerstedt, A. Giua, "Motion probes for fault detection and recovery in networked control systems" *Proceedings of the American Control Conference*, Seattle, WA, USA, June 2008.

[15] S. Kar, J.M.F. Moura, "Distributed Consensus Algorithms in Sensor Networks With Imperfect Communication: Link Failures and Channel Noise". *IEEE Trans. on Signal Processing*, volume 57, pages 355-369, 2009.

[16] A. Kashyap, T. Basar, R. Srikant, "Consensus with Quantized Information Updates" *Proceedings of the 45th IEEE Conference on Decision and Control*, 13-15 Dec. 2006.

[17] R. A. Freeman, P. Yang, K. M. Lynch, "Stability and convergence properties of dynamic average consensus estimators", *Proceedings of the 45th IEEE Conference on Decision and Control*, 13-15 Dec. 2006.

[18] A. Sheth, C. Hartung, R. Han, "A decentralized fault diagnosis system for wireless sensor networks" IEEE International Conference on Mobile Adhoc and Sensor Systems , Nov. 2005.

[19] M.Yu, H.Mokhtar, M.Merabti, "A Survey on Fault Management in Wireless Sensor Networks" School of Computing and Mathematical Science, Proc of PGNET 2007.

[20] G. Lee, T. Tosukhowong, J. H. Lee, Chonghun Han, "Fault Diagnosis Using the Hybrid Method of Signed Digraph and Partial Least Squares with Time Delay: The Pulp Mill Process", Ind. Eng. Chem. Res., 2006.

[21] E. Franco, R. Olfati-Saber, T. Parisini, M.M. Polycarpou, "Distributed Fault Diagnosis using Sensor Networks and Consensus-based Filters", *Proceedings of the 45th IEEE Conference on Decision and Control*, 13-15 Dec. 2006.