Motion Probes for Fault Detection and Recovery in Networked Control Systems

Mauro Franceschelli[⋆], Magnus Egerstedt[†], and Alessandro Giua[⋆]

[⋆]Electrical and Electronic Engineering
University of Cagliari
09123 Cagliari, Italy
{franceschelli,giua}@diee.unica.it

[†]Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA
magnus@ece.gatech.edu

Abstract

In this paper we address the problem of how to excite networked control systems in such a way that faulty, and possibly malicious, agents can be detected. In particular, we envision a scenario in which a subset of the agents in the network are executing a different control strategy than what has been prescribed and the problem under consideration consists of two subtasks, namely detection of the faulty agents, and isolation and mitigation of their impact on the remaining agents. We achieve this by proposing a collection of "motion probes" that leave certain aspects of the network invariant while ensuring that non-cooperative agents are identified and isolated. While we propose a useful tool to identify the faulty agents, the actual algorithms and policies that make use of the motion probes are left to future research.

Published as:
M. Franceschelli, M. Egerstedt, A. Giua, "Motion probes for fault detection and recovery in networked control systems," *2008 American Control Conference*, (Seattle, WA, USA), pp. 4358-4363, June 2008.

I. Introduction

The consensus problem, i.e., the problem of having a collection of agents' states reach a common value in the presence of network and information sharing constraints, has recently received considerable attention. (For a representative sample, see [2], [6], [7], [8], [11], [10], [13].) A common approach to this problem is to use a linear, nearest neighbor control strategy, resulting in a linear dynamic system driven by the *graph Laplacian* associated with the underlying network topology.

One recently discovered aspect of a Laplacian-based control strategy for networked systems is its connection to the heat equation through the introduction of so-called partial difference equations as discrete analogs of partial differential equations [1], [4]. This analogy enables the connection to traditional boundary-value problems. In particular, it has been showed in [5], [12] that the introduction of single anchor nodes, i.e., a single, immobile agent, results in a rendezvous at the location of that agent, provided that the underlying graph remains connected. Similarly, with multiple anchor nodes, the remaining agents converge to the convex hull spanned by the anchor nodes [5].

As a consequence of this, the immobile agents will in effect change the system performance most significantly in that the agents will no longer converge to the centroid of the initial configuration, as would otherwise be the case. Taking this observation one step further, one way in which mobile networks, executing a consensus-based control strategy, can be "hi-jacked" is by either adding a hostile (immobile) agent or by rendering one agent immobile. Moreover, if the hostile agent is moving, it would in essence be able to move all the original agents away from the target area. This fact can be thought of as a rather extreme form of non-robustness with respect to outliers in that outliers are given more or less complete control over the system performance.

In this paper we discuss how to add robustness to the system in the sense that hostile/faulty agents may be identified and their influence nullified. In particular, what we propose is a set of tools for achieving this in a decentralized manner under the banner of so-called *motion probes*. A motion probe is a maneuver executed either by a single agent or by a team of agents, intended to allow the agents to infer certain properties about the network. Moreover, these movements should be such that they preserve desirable properties, such as keeping the centroid static. We point out that such tools can be used as a way of identification of faulty behavior (e.g. if an agent is stuck or is not responding, it is probably faulty) but we do not investigate how to achieve the task in this paper, it will be object of future research. We show however that within this approach it is also possible to recover the original centroid of the non-faulty agents once the faulty agents have been identified.

The outline of this paper is as follows: In Section 2, we briefly discuss the problem under consideration in the setting of linear consensus protocols. Section 3 presents a first result of this paper, namely the motion probes for preserving the rendezvous point (typically the initial centroid of the network) and for nullifying future impacts of hostile/faulty agents. Following this, in Section 4, we present a tool for network fault recovery. It consists of a method to nullify any contribution to the final rendezvous point that the faulty agents may have caused prior to their detection. We conclude the paper with an example and, in Section 5, with the concluding remarks.

II. Problem description

We will be considering the discrete time version of the consensus problem. The results can be extended to continuous time setting in a straightforward manner. A general formulation of the linear, nearest neighbor rule for solving the consensus problem is to let the state of the system evolve 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 $i = 1, \ldots, n$.

We model the network as an undirected graph $\mathcal{G} = V \times E$, with V being a set of vertices $V = \{1, \ldots, n\}$ 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. Based on a matrix representation of this graph, using algebraic graph theory [3], the graph can be encoded through its adjacency matrix A. The adjacency matrix is an $n \times n$ matrix such that $a_{i,j} = 1$ if and only if $(i, j) \in E$ and is 0 elsewhere.

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 $x(k+1) = (I - \epsilon\mathcal{L})x(k)$, where I is the identity matrix, $\mathcal{L} = \Delta - A$ is the graph Laplacian of the graph $\mathcal{G}$, and $\epsilon > 0$. Under this dynamics, the matrix P in Equation (1) becomes

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

Following the notation in [9], we will refer to P as a Perron matrix.

We assume that the topology of the network is represented by a time varying, undirected graph $\mathcal{G}(t) = (V, E(t))$, where the edge set is time dependent, which corresponds to edges being created and disappearing in the network. This could be caused by communication failures, or by the movements of the individual agents as they enter and leave each others' sensory ranges.

Now, as the adjacency and degree matrices are time dependent, the Laplacian will depend on time as well and rather than explicitly computing $\mathcal{L}(t)$, we assume that we have an enumeration of all possible graphs over n agents. We define T as the index set of all connected graphs $T = \{i | \mathcal{G}_i = (V, E_i) \textit{ is connected}\}$. In fact, we will assume that the graph that is currently encoding the network topology is connected, i.e., its index belongs to T, and the linear consensus dynamics that we will employ can thus be given for $k \geq 0$ by $x(k+1) = P_{i(k)}x(k)$ with $x(0) = x_0$, $i(k) \in T$, where x_0 is the initial state of the system. We generalize the dynamics given by (2) assuming that the system matrix $P_{i(k)}$ (corresponding to the connected graph that describes the network topology at time k, i.e., $\mathcal{G}_{i(k)}$) is given by

$$P_{i(k)} = I - \epsilon D\mathcal{L}_{i(k)}, \tag{3}$$

where D is a positive definite, diagonal matrix representing the speed (or gain) of each agent. Obviously for $D = I$ (i.e., all agents have the same speed) equation (3) gives a Perron matrix.

It is straightforward to verify that P_i in Equation (3) is a stochastic, indecomposable, aperiodic matrix for any sufficiently small, positive ϵ. By simple manipulations we find that any $0 \leq \epsilon < \min_j 1/(d_j(n-1))$, where d_j is the j^{th} diagonal entry of D satisfies the above assumption.

III. Motion Probes

In this section we provide the basic tool - the so-called motion probe - for detection of misbehavior in a multi-agent system with single integrator dynamics that is running a linear consensus algorithm.

The formulation of the consensus dynamics in Equation (3) is only valid as long as all agents are executing the prescribed control laws. However, if a subset of the agents do not, the dynamics change and one would typically like to be able to detect this change in dynamics and mitigate its effects. For this, we propose to let individual agents perform controlled movements that is somewhat different from the pure consensus dynamics in order to excite the system. In fact, the main contribution in this paper is the characterization of what movements the agents should perform in order to achieve a variety of tasks like intruder and failure detection, obstacle avoidance or connectivity preservation. Moreover, these motions should be performed in such a way that certain properties of the network are preserved, for instance the centroid of the network (i.e., the rendezvous point) or, in a more general setting, a weighted average of the initial states.

In addition to developing motion probes for exciting the system, we will also provide the tools needed to disconnect an individual agent (typically, the faulty agent) from the network in such a way that the effect that this agent had before it was disconnected can be canceled out. The main application of this technique is the recovery of the initial centroid after the intruders or faulty agents have been detected.

To study the evolution of such systems we need to point out the connection between this formulation of the consensus dynamics and that of discrete time Markov chains. In fact, one can think of P_i as being the transition matrix in a Markov chain, with a corresponding, unique stationary distribution π_i such that $\lim_{k\to\infty} P_i^k = \mathbf{1}\pi_i^T$, where $\mathbf{1}$ is the vector with ones in each entry.

The following result can then be directly obtained:

***Lemma** 1: The stationary distribution of*
$\mathbf{P}(t) = \prod_{k=0}^{t-1} P_{i(k)}$, for any $t \geq 1$, where
$P_{i(k)} = I - \epsilon D \mathcal{L}_{i(k)}$, $i(k) \in T$, is

$$\pi = D^{-1}\mathbf{1}\alpha,$$

with α being a normalizing scalar such that $\sum_{j=1}^{n} \pi_j = 1$.

Proof: The proof follows directly from the basic properties of stochastic, indecomposable, aperiodic matrices. We omit it due to space constraints. ■

Now, what we want is to move a subset of the agents in such a manner that we can infer certain properties of the network. However, at the same time we want to make sure that certain other properties of the network stay the same at the end of the moment. In particular, we will use

Lemma 1 to establish constraints on the movements (or motion probes) that preserve a weighted average of the initial states.

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

$$x(k+1) = P_{i(k)}x(k) + Bu(k) \qquad x(0) = x_0,\ i(k) \in T. \tag{4}$$

Here B is an $n \times n$ matrix (typically the identity matrix), u is a vector of inputs whose i^{th} component u_i is the scalar input exerted by agent i.

***Theorem** 2: Given the network dynamics in Equation* (4). *If*

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

then $\pi^T x(t) = \pi^T x(0)$, where π is the stationary distribution in Lemma 1.[1]

Proof: We have that

$$x(t) = \prod_{k=0}^{t-1} P_{i(k)}x(0) + \sum_{k=0}^{t-1}\prod_{j=0}^{k} P_{i(j)}Bu(k).$$

Multiplying by π^T on both sides of the above equation and observing that π is the stationary distribution of $P_i\ \forall i \in T$, we get

$$\pi^T x(t) = \pi^T x(0) + \sum_{k=0}^{t-1} \pi^T Bu(k).$$

Since $\pi^T B$ does not depend on k it can be taken out of the summation as

$$\pi^T x(t) = \pi^T x(0) + \pi^T B \sum_{k=0}^{t-1} u(k).$$

By hypothesis, $\sum_{k=0}^{t-1} u(k) = 0$, and hence $\pi^T x(t) = \pi^T x(0)$, which concludes the proof. ■

The previous theorem can be understood in the context of the partial difference equation analogy with the heat equation. Any agent applying an input can be seen as an agent that is "warming up" or "cooling down" the network, depending on the sign of the input. Since the system is conservative (no heat can flow away), in order to recover the initial thermal equilibrium point the only information needed is how much heat has flown in or out from the network. This quantity corresponds to the integral of the applied input. Hence, if the integral is zero, the total heat present in the network has been preserved, and the initial, thermal equilibrium point will be reached under the regular evolution of the heat equation.

It should moreover be pointed out that such a motion probe can be performed by any agent in a completely decentralized fashion since no information is required to flow through the network

[1]If the n agents are moving in a $m-$dimensional space, the extension to $\mathbb{R}^{n\times m}$ requires the presence of a relative inertial reference for each agent. This means that the assumption that $\sum_{k=0}^{t-1} u(k) = 0$ needs to hold for $u \in \mathbb{R}^{n\times m}$.

for its computation. In other words, the motion probe can be used to achieve a variety of tasks like *obstacle avoidance*, *failure detection* (i.e., if an agent does not react to the motion probe it is clearly not running the consensus algorithm), and *connectivity preservation* (an agent may just slow down to avoid disconnection with a far agent and then speed up later to preserve the centroid). And, this is done while preserving the weighted centroid of the network, as per Theorem 2.

In particular, we envision the motion probe to be used in the following scenarios:

1) Agents are Rendered Immobile: If reliable communications are not available, or costly, we can imagine that the neighbors of the immobile agent can adopt a policy used to identify the faulty behavior of the immobile agent. This can be achieved by checking its reaction to the motion probes.

2) Motions as Communication: In a swarm of agents in a futuristic scenario the leaders may give simple orders by performing some movements that will be repeated by the neighbors and so on to transfer information without disrupting certain properties of the network. In other words, the motion probe leaves the centroid intact while serving as means of propagating information through the network.

3) Non-Consensus Based Control: Given an initial configuration of the network. If all the motions are composed entirely by motion probes, the centroid will remain static, allowing for a much more expressive set of maneuvers.

IV. Fault recovery

If we assume that we have been able to locate a faulty agent (perhaps using the motion probe discussed in the previous section), what we would like to do is to isolate that agent from the network. Moreover, we would like to not only cancel out that agent's effect on the system after recovery, but also to nullify the agents total effect, from time $t = 0$ and 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 agent 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 will let the network topology be static in the following paragraphs. All the computations will still hold under slightly more general assumptions that we briefly mention later. The general case of switching topology will be the object of future research. We assume that the agents are ordered in such a way that the first $n - m$ agents are non-faulty, and the remaining m agents are faulty. In fact, we let the system dynamics be given by

$$x(k+1) = \hat{P}x(k) + \hat{B}u(k) \qquad x(0) = x_0. \tag{5}$$

Here $\hat{B}$ is $\hat{B} = \left[\begin{array}{c|c} B_g & \mathbf{0}_{n-m\times m} \\ \hline \mathbf{0}_{m\times n-m} & B_f \end{array}\right]$, where B_g and B_f are the input matrices of respectively the non-faulty and the faulty agents and B_g corresponds to the identity matrix with the appropriate dimensions. B_g is assumed to be the identity matrix since this system represent a collection of agents that though collaborating to achieve a common goal need to perform some tasks on their own (i.e., motion probes or else). However, $\hat{P}$ is no longer a Perron matrix.

$\hat{P}$ can be expressed as follows: As in the previous section, we let D denote the positive definite, diagonal weight (or gain) matrix associated with each agent. Moreover, 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 $\hat{P} = I - \epsilon D(\mathcal{L} - W_f\mathcal{L})$, where $\mathcal{L}$ is the graph Laplacian.

It is straightforward to show that $\hat{P}$ in Equation (5) can be written as the following, partitioned block matrix:

$$\hat{P} = \left[\begin{array}{c|c} P_g & D_f \\ \hline \mathbf{0}_{m\times n-m} & I_{m\times m} \end{array}\right] \tag{6}$$

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

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

$$\begin{aligned} x_g(k+1) &= P_g x_g(k) + D_f x_f(k) + B_g u_g(k) \\ x_f(k+1) &= x_f(k) + B_f u_f(k). \end{aligned} \tag{7}$$

Viewed at the level of the individual non-faulty agents, the dynamics of agent i is in fact given by

$$x_i(k+1) = x_i(k) - \epsilon d_i \sum_{j\in\mathcal{N}_i}(x_i - x_j) + u_i(k),$$

where, as before, $\mathcal{N}_i$ is the set of vertices adjacent to vertex i. Letting $F = \{i \mid \text{ agent } i \text{ is faulty}\}$ allows us to separate the contributions to the non-faulty agent's evolution as

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

We may call P_s the matrix that describes the dynamic of the good agents and see as input the distance between the faulty agents and their neighbors and corresponds to $P_s = I - \epsilon D\mathcal{L}_i$ where $\mathcal{L}_i$ is the subgraph induced by the good agents. With simple algebraic manipulations one may note that $P_s = P_g + H$ where H is a $n-m\times n-m$ diagonal matrix whose i^{th} element corresponds to the number of faulty agents in the neighborhood of agent i multiplied by ϵd_i: $H = \epsilon D \text{diag}\left(\begin{array}{ccc} |\mathcal{N}_1 \cap F| & \cdots & |\mathcal{N}_{n-m} \cap F| \end{array}\right)$.

The following theorem provides a tool for network recovery after some agents 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. The theorem is stated for a fixed connected topology and then the result is extended to the case of a switching topology.

***Theorem** 3: Let the network of agents be described by Equation* (7)*. If the following conditions hold:*

1) *At time t_0 all m faulty agents have been detected.*
2) *The neighbors of faulty agents apply a control input 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 agents has been performed, i.e.,

$$\pi^T x_g(0) = \pi^T x_g(t).$$

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

$$\lim_{k\to\infty} x_g(t+k) = \mathbf{1}\pi^T x_g(0).$$

Proof: The states of the non-faulty agents follow the equation $x_g(k+1) = P_g x_g(k) + D_f x_f(k) + B_g u_g(k)$. What each non-faulty agent can actually measure, in the proposed discrete time consensus algorithm, is the difference between its own state and all the neighbors'. The generic neighbor i of a faulty agent j sees as input $N_{ij}(k) = e_i^T(D_f e_j e_j^T x_f(k) - H x_g(k))$ and can easily keep the information about the total contribution from each neighbor by remembering $\sum_{k=0}^{t_0-1} N_{ij}(k)$. The evolution of the system can now be described by the following equations:

$$\begin{aligned} x_g(k+1) &= P_s x_g(k) + D_f x_f(k) - H x_g(k) + B_g u_g(k) \\ x_f(k+1) &= x_f(k) + B_f u_f(k). \end{aligned}$$

We recall that $P_s = P_g + H$ is a stochastic, indecomposable, aperiodic matrix by construction since $P_s = I - \epsilon D \mathcal{L}_s(\mathcal{G})$, where $\mathcal{L}_s(\mathcal{G})$ is the Laplacian of the induced subgraph of the non-faulty agents (that is assumed to be connected). The states of the good agents at time t_0 as function of the faulty agents and their initial state can be written as:

$$x_g(t_0) = P_s^{t_0} x_g(0) + \sum_{k=0}^{t_0-1} P_s^k N(k) + \sum_{k=0}^{t_0-1} P_s^k B_g u_g(k).$$

At time t_0 all the faulty agents are detected and each neighbor of a faulty node is disconnected. Then, after an arbitrary number of time steps, each agent that was a neighbor of a faulty one applies the proposed control input based on the information preserved locally about the contribution of the faulty neighbor to its dynamic. For this, let $\widehat{u_g} = u_g + u_{rec}$ be the non-faulty agents' inputs between time t_0 and t, where u_g is any input that the non-faulty (good) agents may want to perform (i.e., a motion probe or else) and u_{rec} is the recovery input. When, at time t, each agent stops applying the recovery input, the state of the network is

$$\begin{aligned} x_g(t) &= P_s^{t-t_0}(P_s^{t_0} x_g(0) + \sum_{k=0}^{t_0-1} P_s^k N(k) \\ &+ \sum_{k=0}^{t_0-1} P_s^k u_g(k)) + \sum_{k=t_0}^{t-1} P_s^k B_g \widehat{u_g}(k). \end{aligned}$$

Multiplying both sides of the above equation by the stationary distribution of P_s, π^T:

$$\begin{aligned} \pi^T x_g(t) &= \pi^T P_s^{t-t_0}(P_s^{t_0} x_g(0) + \sum_{k=0}^{t_0-1} P_s^k N(k) \\ &+ \sum_{k=0}^{t_0-1} P_s^k B_g u_g(k)) + \pi^T \sum_{k=t_0}^{t-1} P_s^k B_g \widehat{u_g}(k), \end{aligned}$$

since $\pi^T P_s^k = \pi^T$, $\forall k \geq 0$, we get

$$\begin{aligned} \pi^T x_g(t) &= \pi^T x_g(0) + \pi^T \sum_{k=0}^{t_0-1} N(k) + \pi^T B_g \sum_{k=0}^{t_0-1} u_g(k) \\ &+ \pi^T B_g \sum_{k=t_0}^{t-1} \widehat{u_g}(k). \end{aligned}$$

Recalling that B_g is the identity matrix and $\widehat{u_g} = u_g + u_{rec}$ gives

$$\begin{aligned} \pi^T x_g(t) &= \pi^T x_g(0) + \pi^T \sum_{k=0}^{t_0-1} N(k) + \pi^T \sum_{k=0}^{t_0-1} u_g(k) \\ &+ \pi^T \sum_{k=t_0}^{t-1} u_g(k) + \pi^T \sum_{k=t_0}^{t-1} u_{rec}(k). \end{aligned}$$

Since $\sum_{k=t_0}^{t} u_{rec}(k) = -\sum_{k=0}^{t_0-1} N(k)$, we get

$$\pi^T x_g(t) = \pi^T x_g(0) + \pi^T \sum_{k=0}^{t-1} u_g(k).$$

Then, if u_g is identically null or is the resulting of tasks accomplished with motion probes we know that

$$\sum_{k=0}^{\tau-1} u(k) = 0, \quad \forall \tau \geq t_0,$$

and thus $\pi^T x_g(t) = \pi^T x_g(0)$, thus proving the first part of the theorem.

If furthermore $\mathcal{G}$ stays connected after all the faulty agents have been removed from the network, the hypothesis for the consensus algorithm to converge are satisfied and:

$$\lim_{k \to \infty} x_g(k) = \mathbf{1} \pi^T x_g(0),$$

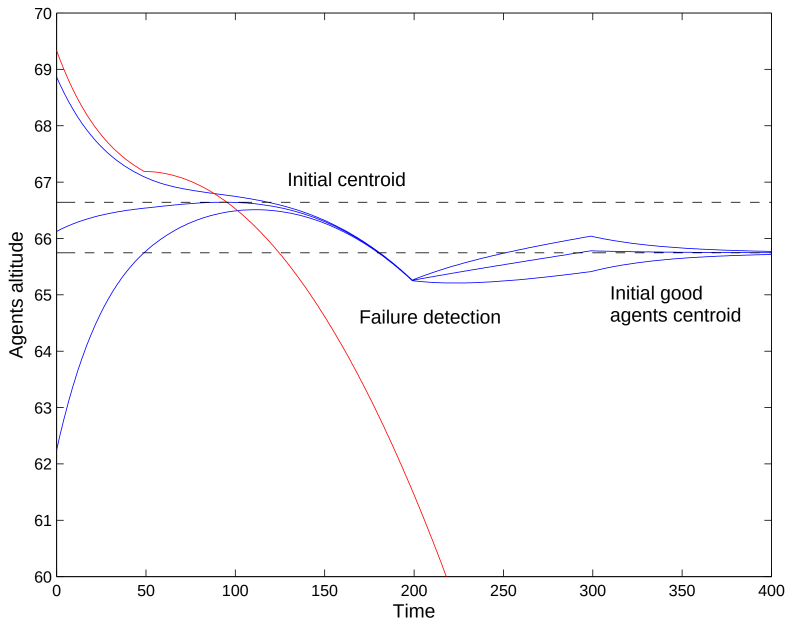

Fig. 1. Evolution of the network of agents in Example 1. Four UAVs are tasked with flying at the same height while avoiding to be dragged to the ground when one is shot down.

and the second part of the theorem follows. ■

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

1) Only the agents that are neighbors to a faulty agent need to apply the recovery input.
2) The information needed by the generic agent 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 agent.
3) Any agent that has detected a faulty agent in its neighborhood may apply the recovery at any time. There is no need that all the faulty agents are detected at the same time. The recovery procedure may then be applied in an asynchronous way.

The extension of the previous theorem to the switched topology case will be object of future research. We point out that a trivial sufficient condition for Theorem 3 to hold in the switching topology case is that the set of neighbors of the faulty agents do not decrease in number. In such case any agent need only to remember the inputs of its actual neighbors.

Next we give an example of fault recovery for a simplified case of a network of unmanned aerial vehicles. These UAVs are assumed to be tasked with flying at the same height while avoiding be dragged to the ground when one of them is shot down. These agents are moreover assumed to be aware only of the relative distances of the neighbors.

***Example** 1:* Let the evolution of the network of agents in Figure 2, be described by $x(k+1) = Px(k)$, where

$$P = I - \epsilon \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}$$

and $x(0) = [66.12\ 62.24\ 68.86\ 69.33]^T$. The rendezvous point is then at 66.64 and we note that the average of the states of the non-faulty agents $1, 2$ and 3 is 65.74 at the initial time. The evolution of such network is shown in Figure 1. The agents perform the consensus algorithm on their altitude when at $k = 50$ agent 4 is shot down and starts falling to the ground as shown in Figure 1. The other agents, still unaware of what happened try to follow his movements, being dragged to the ground themselves. The dynamic of the network during this period is described by:

$$\begin{aligned} x_g(k+1) &= \left(I - \epsilon \begin{bmatrix} 3 & -1 & -1 \\ -1 & 3 & -1 \\ -1 & -1 & 3 \end{bmatrix} \right) x_g(k) + \epsilon \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} x_f(k) \\ x_f(k+1) &= x_f(k) + u(k). \end{aligned}$$

Then at time $k = 200$ by some means the neighbors of the broken agent realize that agent 4 was broken (i.e., identify the suspicious behavior and execute a motion probe or through other means achieve the same result). Agent 4 is then disconnected from the neighbors, the dynamic of the network then becomes:

$$x_g(k+1) = \left(I - \epsilon \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \right) x_g(k) + \epsilon \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} u(k)$$

Where $u(k)$ is the recovery input. In this example all the agents detect the misbehavior, disconnect the faulty agent and apply the recovery input at the same instant of time for clarity sake. All this operations may take place at different instants of time in a real application. The agents know the summation of inputs due to agent 4:

$$\begin{aligned} &Agent\ 1: && \textstyle\sum_{i=0}^{k-1} x_1(i) - x_4(i) = 72.8, \\ &Agent\ 2: && \textstyle\sum_{i=0}^{k-1} x_2(i) - x_4(i) = -48.2, \\ &Agent\ 3: && \textstyle\sum_{i=0}^{k-1} x_2(i) - x_4(i) = 158.42. \end{aligned}$$

So all of them need to apply an input whose summation over a finite number of steps is opposite to the one applied by the broken agent. For simplicity the chosen input is constant with a length of 100 time steps (i.e., the agents are completely free to do whatever they like as long as the total contribution of agent 4 is nullified). Such inputs for agent $1, 2$ and 3 are:

$$\begin{aligned} u_{rec,1}(k) &= -0.72, && k = 201, \ldots, 300 \\ u_{rec,2}(k) &= 0.48, && k = 201, \ldots, 300 \\ u_{rec,3}(k) &= 1.58, && k = 201, \ldots, 300 \end{aligned}$$

and zero elsewhere.

When they all finish applying the recovery at time 300, we note in Figure 1 that the average of their altitudes has become exactly their average at the initial instant of time, namely 65.74. From this point on the network evolves as a standard consensus network, reaching a practical rendezvous around time $k = 400$. ■

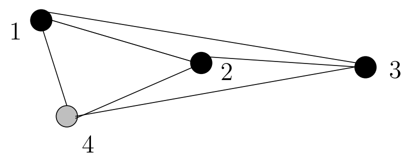

Fig. 2. Network of agents in Example 1.

V. Conclusions

In this paper we consider the problem of how to move agents in a networked system in such a way that they do not change the desired rendezvous point during the maneuver. Such movements are referred to as motion probes. And, in particular, we show how such motion probes could be used to identify faulty agents that do not exhibit the prescribed dynamical behavior. Moreover, once these faulty agents have been identified, we show how to nullify their impact on the behavior of the non-faulty agents. We also outline some particularly promising directions for future research in this new area of motion probes for exciting networked control systems.

Acknowledgments

The work by Magnus Egerstedt was supported in part by the U.S. Army Research Office through Grant # 99838.

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.