2012 American Control Conference
Fairmont Queen Elizabeth, Montréal, Canada
June 27-June 29, 2012

Distributed Real-time Fault Detection and Isolation For Cooperative Multi-agent Systems

Meng Guo, Dimos V. Dimarogonas and Karl Henrik Johansson

Abstract— **In this paper we propose a distributed real-time fault detection, isolation and mitigation framework for multi-agent systems performing cooperative tasks. Various system models and detection schemes with respect to communication and sensing are considered. Two communication protocols for fault detection are introduced first and proved to be effective. Then a scheme based on limited relative state measurements is developed. Furthermore, we propose fault isolation and mitigation steps to guarantee the accomplishment of a global objective. All schemes are distributed in the sense that at each step of the fault detection, isolation and mitigation every agent only uses locally available information. One key feature of the framework is the significant reduction of required computational resource when compared with the fault detection and isolation schemes based on unknown input observers. Later we show that the proposed framework can be applied to the consensus and other cooperative formation problems. Several computer simulations are presented to demonstrate the efficiency of the framework.**

I. INTRODUCTION

A large number of multi-agent applications focuses on achieving cooperative global objectives using distributed control laws, like consensus [10], formation control [1], and flocking [15]. On the one hand, decentralized structures are suitable for large systems due to their scalability with respect to the number of agents and the flexibility and complexity a system can achieve. On the other hand, the lack of a central authority that monitors the activity of the nodes of the network renders the distributed system vulnerable to malfunctions and attacks. For example, an immobile agent in the group performing formation control could jeopardize the performance significantly, such that all agents could get stuck around this node. In the case of malicious behaviors, the team is at risk of being led to any final configuration desired by the hostile agents. Similar potential hazards can be traced in many other cooperative control scenarios.

Thus it is of vital importance to add resilience to the system in the sense that non-cooperative behaviors should be detected and malicious or faulty agents should be isolated from the group. Many related results have appeared and the existing approaches can be separated into two categories. The first scheme proposed in [13] and [14] is to use Unknown Input Observers (UIO), where after a sufficiently large number of time-steps each node is expected to have enough information to calculate the desired equilibrium point by estimating the initial states. Conditions on the connectivity of the graph to correctly estimate the initial values in the presence of faulty agents are obtained using properties of the *observability matrix* and *fault matrix* [11]. Faulty node removal algorithms with or without external inputs are presented in [13]. Instead of passive state estimation, [7] provides the scheme to excite networked control systems with additional excitation signals so that faults can be detected, using the so-called *motion probes*. The fault detection step is followed by the fault isolation and mitigation of the faulty agents' impact on the remaining agents. Heuristic test signals like square waves are introduced and the rendezvous point is proved to be preserved. Some other fault tolerance techniques designed for certain applications can be found in [12] where faulty agents are assigned higher priorities to be always visible to other participants, and in [13] where the existence of UIO for networks of interconnected second-order systems is studied. The idea of *local monitors* is first introduced in [4] and [6] where each agent is responsible for monitoring all neighboring agents that lie within a safety region. The cooperation rules are assumed to be a class of decentralized logical conditions. Moreover, a communication-based reputation consensus protocol is constructed in [5] in order to confirm the network's decisions on faulty targets.

In this paper, we propose a real-time distributed fault detection framework that requires less computational resources than UIO based methods and is easer to implement with respect to motion-probes based approaches. Moreover, it is easily applicable to many multi-agent cooperative control scenarios. The main contributions of this work are: (i) a formal definition of faults in a multi-agent system, (ii) a fault detection framework in which each node monitors its neighbors by using only local information, (iii) two communication protocols for detecting potential faults at different locations, (iv) a new network structure for multi-agent systems with on-board relative state measuring sensors, (v) a fault isolation and mitigation scheme to guarantee the global performance, which is robust with respect to model uncertainties and disturbances.

The rest of the paper is organized as follows: the system model and problem descriptions will be stated in Section II. We introduce our fault detection schemes and prove their effectiveness under different system models in Section III. Section IV is devoted to the discussion of fault isolation and system recovery after certain faulty behaviors are detected while Section V addresses the possible applications to multi-agent consensus and formation problems. Computer simulations to support the results can be found in Section VI.

The authors are with the ACCESS Linnaeus Center, School of Electrical Engineering, KTH Royal Institute of Technology, SE-100 44, Stockholm, Sweden. mengg, dimos, kallej@kth.se. This work was supported by the Swedish Research Council (VR), and the Knut and Alice Wallenberg Foundation. The second author is also affiliated with the KTH Centre for Autonomous Systems and is supported by VR through contract 2009-3948.

978-1-4577-1094-0/12/$26.00 ©2012 AACC

II. Model Description

In this section, we first present the agent dynamics under consideration and then the communication and sensing based system models. A formal definition of faulty agents and the problem statement are then provided.

A. Agent Dynamics

Consider a sampled model of single integrator agents:

$$z_i((k+1)T) = z_i(kT) + u_i(kT)T, \quad i = 1, \cdots, N, \quad (1)$$

where T is the sampling time. For brevity, we denote $z_i^k = z_i(kT)$ and $u_i^k = u_i(kT)$, $\forall k \in \mathbb{Z}^+$. $z_i^k = [x_i^k, y_i^k]^T \in \mathbb{R}^2$ is the position coordinate in the 2-D configuration space and $u_i^k \in \mathbb{R}^2$ the control input of agent i at time step k. Multi-agent cooperation is achieved on the basis of information exchange among the group, either by means of direct wire or wireless communication or perception with on-board sensors. The inter-agent communication or sensing network can be encoded in terms of undirected graphs $G = \{V, E\}$, which consist of a set of vertices $V = \{1, \cdots, N\}$ indexed by the the team members and a set of edges $E \subset V \times V$. We define that agent i and j are neighbors and namely $(i, j) \in E$ when they have communications with each other or they stay in each other's sensing zone. Let $\mathcal{N}_i^k$ be the set of agent i's neighbors at time step k and $|\mathcal{N}_i^k|$ denotes its cardinality. In the sequel, we always assume that G is undirected, i.e., $(i, j) \in E \Leftrightarrow (j, i) \in E$.

We further assume that the distributed control law of (1) has the following structure:

$$u_i^k = \mathcal{P}_i(z_i^k, I_i^k), \quad (2)$$

where the function $\mathcal{P}_i : \mathbb{R}^2 \to \mathbb{R}^2$ prescribes the control objective, $I_i^k = \{z_{i_1}^k, \cdots, z_{i_p}^k\}$ is the set of agent i's neighbors' states at time step k, where $\mathcal{N}_i^k = \{i_1, \cdots, i_p\}$ and $p = |\mathcal{N}_i^k|$. The form of (2) is frequently encountered in multi-agent cooperative controllers. Moreover, we denote by $\mathcal{P} = \{\mathcal{P}_1, \cdots, \mathcal{P}_N\}$ the set of considered cooperative protocols. $\mathcal{P}$ is called homogeneous if $\mathcal{P}_i = \mathcal{P}_j$, $\forall (i, j) \in V \times V$, otherwise it is non-homogeneous.

B. Communication-based Model

We will first assume that each agent's perception of the environment is communication based. We refer to this model as the *communication-based* model, where $\mathcal{N}_i^k$ is defined as the subset of agents with which agent i communicates at time step k. We consider here two protocols designed for the *communication-based* model to perform fault detection, the effectiveness of which will be analyzed in Section III-A.

Protocol I: Agent $i \in V$ transmits the value of u_i^k which is computed by (2), along with its absolute state z_i^k to all its neighbors $j \in \mathcal{N}_i^k$ at each time step k.

Protocol II: Agent $i \in V$ transmits its absolute state z_i^k and all the elements in the set I_i^k to all its neighbors $j \in \mathcal{N}_i^k$ at each time step k.

As a result, the information or message any agent $i \in V$ receives at time step k is the set $m_i^k \subset \mathbb{R}^2$, which is given by $m_i^k = \{z_j^k, u_j^k\}$, $\forall j \in \mathcal{N}_i^k$ in *Protocol I* and $m_i^k = \{z_j^k, I_j^k\}$, $\forall j \in \mathcal{N}_i^k$ in *Protocol II.*

C. Sensing-based Model

Another practical way of environment perception is based on taking measurements from on-board sensors. In particular, we assume each agent is capable of measuring the relative states between itself and agents within a certain range $R \in \mathbb{R}^+$. Namely, the measurement set is given as $z_i^k - I_i^k = \{z_i^k - z_{i_1}^k, \cdots, z_i^k - z_{i_p}^k\}$ and $\mathcal{N}_i^k = \{i_1, \cdots, i_p\} = \{j \in V | \, 0 < \|z_i^k - z_j^k\| \leq R\}$. Thus the control law (2) is implemented as $u_i^k = \mathcal{P}_i(z_i^k, I_i^k) = \mathcal{P}_i(z_i^k, z_i^k - (z_i^k - I_i^k))$. We refer to this type of model as the *sensing-based* model. A special designed sensing network structure for sensing-based models is introduced below:

Control Network: This network is intended for the control objective and it is defined as $\mathcal{N}_i^{c,k} = \{j \in V | \, 0 < \|z_i^k - z_j^k\| \leq r_c\}$, where $0 < r_c < R$ and $\mathcal{N}_i^{c,k}$ is called the *control neighboring set* at time step k. r_c is the control range and $\{z \in \mathbb{R}^2 | \|z_i^k - z\| \leq r_c\}$ stands for the control zone. Thus the control protocol $\mathcal{P}$ is implemented as $u_i^k = \mathcal{P}_i(z_i^k, I_i^{c,k})$, where $I_i^{c,k} = \{z_{i_1}^k, \cdots, z_{i_p}^k\}$ and $\mathcal{N}_i^{c,k} = \{i_1, \cdots, i_p\}$.

Fault Detection Network: This network is used for the purpose of fault detection. We have $\mathcal{N}_i^{f,k} = \{j \in V | \, 0 < \|z_i^k - z_j^k\| \leq r_f\}$, where $0 < r_f < R$ and $\mathcal{N}_i^{f,k}$ is called the *fault detection neighboring set* at time step k. r_f stands for the detection range and $\{z \in \mathbb{R}^2 | \|z_i^k - z\| \leq r_f\}$ is called the detection zone.

In this case, any agent $i \in V$ has two different set of measurements at time step k, i.e., $m_i^{c,k} = \{z_i^k - I_i^{c,k}\}$ and $m_i^{f,k} = \{z_i^k - I_i^{f,k}\}$, obtained by detecting the control network and fault detection network.

D. Problem statement

The intuitive idea of the *model-based* fault diagnosis technique introduced in [3] is to reconstruct the process behavior on-line by creating either hardware redundancy or software redundancy. The process model will run in parallel to the real process and driven by the same inputs. It is expected that they share the same process variables otherwise the difference between the reconstructed process variables and actual measurements will be recorded as *residual signals*.

In our multi-agent system, each agent is treated as a process and needs an individual fault detection system. Since it is of considerable interest to monitor the consecutive displacement $z_i^{k+1} - z_i^k$, or namely the velocity input u_i^k of the agents, we use u_i^k as the performance index [3] to generate the residual $r_i^k = u_i^{r,k} - u_i^{a,k}$, where $u_i^{r,k} \in \mathbb{R}^2$ is the reasonable movement at time step k based on the cooperation law $\mathcal{P}$ and $u_i^{a,k} \in \mathbb{R}^2$ is the actual movement estimated from real-time measurements. We will specify how $u_i^{r,k}$ and $u_i^{a,k}$ are defined for each of the models in Section III. We firstly introduce the following definition of faulty agents:

Definition 1: Let us consider the system (1) under control laws (2). One agent i participating in the cooperative task described above is classified as faulty at time step k if

$$\|r_i^k\| = \|u_i^{r,k} - u_i^{a,k}\| > \chi(\|u_i^{r,k}\|, \delta) \quad (3)$$

where $\chi(\|u_i^{r,k}\|, \delta)$ is a threshold function, depending on the input signal norm $\|u_i^{r,k}\|$ and the disturbance δ.

One possible structure of $\chi(\|u_i^{r,k}\|, \delta)$ given in [3] is $\chi(\|u_i^{r,k}\|, \delta) = \gamma_1 + \gamma_2 \|u_i^{r,k}\|$, in which the constant part γ_1 depends on disturbance δ, while the time varying part $\gamma_2 \|u_i^{r,k}\|$ is related to the instantaneous energy of the input.

Our goal for the system involving possible faulty agents is to accomplish the desired global objective while at the same time detecting and isolating these faulty agents. Since those faulty agents are unable to participate in the global task, we consider the task accomplished when the non-faulty agents fulfill their parts of the global objective.

III. Distributed Fault Detection

Most of the existing approaches mentioned previously require every participating member to estimate the current state or initial value of the whole system to observe possible unknown inputs at any other node. This is not only computationally expensive for local controllers, but also not scalable to large systems. Motivated by this observation, we believe that it is intuitive and reasonable to rely on neighboring agents to monitor each other's behavior, rather than those connected by long paths, which will inherently lead to a distributed fault detection framework. Moreover we make the assumption that every agent in the group should participate and perform the fault detection and isolation mechanism. Before introducing our main fault detection framework, we establish the following definition:

Definition 2: Given a multi-agent system as described above, the objective of our fault detection framework is achieved if every non-faulty agent successfully detects possible faulty agents belonging to its (i) communication set for *communication-based* models, (ii) control neighboring set for *sensing-based* models.

In our fault detection framework, every agent acts as a local monitor that monitors the behavior of its neighboring agents. Thus every agent needs to generate a residual signal for every one of its neighbors, i.e., agent $i \in V$ has to generate a residual signal r_j^k for each $j \in \mathcal{N}_i^k$ at time step k.

In order to generate and evaluate the residual signal r_j^k based on (3), we need to determine two terms: $u_j^{a,k}$ and $u_j^{r,k}$. The first term is estimated by $u_j^{a,k} = h(z_j^{k+1}, z_j^k)$, if agent j's consecutive states z_j^{k+1} and z_j^k are available. The function $h : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}^2$ can be simply first oder differentiation $(z_j^{k+1} - z_j^k)/[(k+1)T - kT]$ or combined with other post-processing techniques.

So the main challenge of agent i acting as a local monitor is to reconstruct $u_j^{r,k}$ correctly for each $j \in \mathcal{N}_i^k$. Due to the limited sensing range and communication constraints, agent i may not have full access to the control elements of agent j, i.e., the state of agent j's neighbors I_j^k. In the following, we will introduce different approaches to reconstruct $u_j^{r,k}$ under the *communication-based* or *sensing-based* models.

A. Communication-based Fault Detection

Given the *communication-based* models, possible faults in different locations can be identified by taking advantage of *Protocol I* and *Protocol II*.

To make the idea more clear, as shown in Fig. 1, we divide the process, from when agent j receives the information m_j^k from its neighbors $l \in \mathcal{N}_j^k$ to when it actually moves accordingly from z_j^k to z_j^{k+1}, into two phases. In phase one, agent j computes the reasonable control input $u_j^{r,k}$ according to the protocol $\mathcal{P}$ on its digital platform. In phase two, the derived $u_j^{r,k}$ is relayed and transmitted through electrical or mechanical components to the actuation parts like motors, leading to agent j's actual movement $u_j^{a,k}$, which is in turn directly estimated as $h(z_j^{k+1}, z_j^k)$ by agent $i \in \mathcal{N}_j^k$.

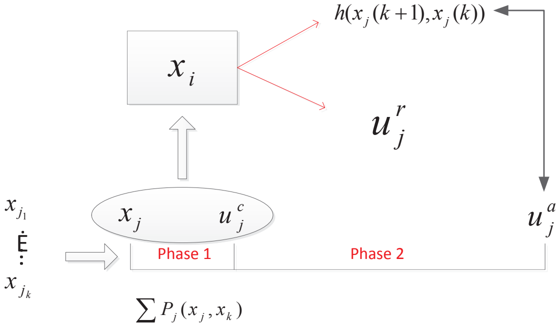

Fig. 1. Communication-based detection

Since phase one only involves the digital implementation of the protocol $\mathcal{P}$ and phase two may contain mechanical transmission hazards or inaccurate actuation models, we first tackle the case where $u_j^{r,k}$ is assumed to be correctly computed in phase one and possible bias and malfunctions are introduced in phase two.

Theorem 1: Faults in phase two can be detected in real-time for *communication-based* models using *Protocol I*.

Proof: By applying *Protocol I*, agent i can generate the residual signal r_j^k easily for each $j \in \mathcal{N}_i^k$. This is because $r_j^k = u_j^{a,k} - u_j^{r,k} = h(z_j^{k+1}, z_j^k) - u_j^{r,k}$, where z_j^{k+1}, z_j^k and $u_j^{r,k}$ are all obtained directly from m_i^{k+1} and m_i^k through the definition of *Protocol I*. Given the suitable threshold function $\chi(\|u_j^{r,k}\|, \delta)$ in (3), potential faults appearing in phase two of agent $j \in \mathcal{N}_i^k$ can be identified in real-time by agent i. Note that the approach described above can be applied to the group of agents performing different tasks, which means $\mathcal{P}_i$ is not necessarily the same as $\mathcal{P}_j$, $\forall j \in V$. ■

On the other hand, when faults are found in phase one, in the sense that either agent j is unable to compute $u_j^{r,k}$ correctly based on the protocol $\mathcal{P}$ or it maliciously transmits a *fake* $u_j^{r,k}$ to its neighbors and acts accordingly, then the previous approach becomes infeasible because now $u_j^{r,k}$ and $u_j^{a,k}$ are nearly identical if there are no faults in phase two. This poses some difficulty since now agent i needs full state information of both agent j and agent j's neighbors, namely z_j^k and I_j^k, to reconstruct $u_j^{r,k}$ by $\mathcal{P}_j(z_j^k, I_j^k)$.

Theorem 2: Faults in phase one and two can be detected in real-time for *communication-based* models using *Protocol II*, if $\mathcal{P}$ is homogeneous, i.e., $\mathcal{P}_i = \mathcal{P}_j$, $\forall (i, j) \in V \times V$.

Proof: According to *Protocol II*, agent j transmits the absolute state of itself and its neighbors: z_j^k and I_j^k to all its neighbors including agent i. Therefore, agent i can reconstruct $u_j^{r,k}$ precisely by $\mathcal{P}_j(z_j^k,\, I_j^k))$, using the received information $\{z_j^k,\, I_j^k\} \subset m_i^k$ and the homogeneous function $\mathcal{P}_j = \mathcal{P}_i$. Then agent i can generate the residual signal $r_j^k = h(z_j^{k+1},\, z_j^k) - u_j^{r,k}$ for each $j \in \mathcal{N}_i^k$, using received information m_i^{k+1} and m_i^k. Given the suitable threshold function $\chi(\|u_j^{r,k}\|, \delta)$ in (3), every agent in the group is capable of detecting possible faults in its neighborhood. Note that we require $\mathcal{P}$ to be homogeneous here so that agent i can reconstruct $u_j^{r,k}$ using its own control law $\mathcal{P}_i$. ■

It is worth mentioning that *Protocol I* does not increase the communication load of the whole system too much as only one extra real number is added for one neighbor, while more information exchange is demanded in *Protocol II* especially with strongly connected and large graphs. However, since the amount of extra transmission depends on the number of communication links in the network, we can reduce it by limiting the number of neighbors each agent is allowed to communicate with.

B. Sensing-based Fault Detection

We now consider sensing-based models in which all agents have on-board sensors with the same control range r_c and fault detecting range r_f. Furthermore, We assume that each agent has at least one neighbor and the group of faulty agents is connected to at least one non-faulty agent.

Following the definitions of control network and fault detection network in Section II-C, we can establish an important relation between $\mathcal{N}_i^{c,k}$ and $\mathcal{N}_i^{f,k}$ at time step k.

Lemma 3: Given that $0 < 2r_c < r_f \leq R$ holds, the following relation is guaranteed for the *sensing-based* models: $j \in \mathcal{N}_i^{c,k} \;\wedge\; l \in \mathcal{N}_j^{c,k} \;\wedge (i \neq l) \Rightarrow l \in \mathcal{N}_i^{f,k}$.

Proof: We choose $j \in \mathcal{N}_i^{c,k}$ and $l \in \mathcal{N}_j^{c,k}$ in the control zone. Since $2r_c < r_f \leq R$, we have $|z_i^k - z_l^k| = |z_i^k - z_j^k + z_j^k - z_l^k| \leq |z_i^k - z_j^k| + |z_j^k - z_l^k| \leq r_c + r_c = 2r_c < r_f$, which means agent l must belong to agent i's fault detection zone. We exclude the case $i = l$ since $i \in \mathcal{N}_j^{c,k}$ but $i \notin \mathcal{N}_i^{f,k}$ by definition. Thus the neighbors of agent i's neighbors in the control zone must belong to agent i's fault detection zone. ■

Theorem 4: When $0 < 2r_c < r_f \leq R$ holds, possible faults in the *sensing-based* models can be detected in real-time, if $\mathcal{P}$ is homogeneous, i.e., $\mathcal{P}_i = \mathcal{P}_j$, $\forall (i,\, j) \in V \times V$.

Proof: We continue the discussion of generating the residual signal $r_j^k = u_j^{r,k} - u_j^{a,k}$. Agent $i \in \mathcal{N}_j^{c,k}$ can estimate the actual movement $u_j^{a,k} = h(z_j^{k+1},\, z_j^k)$ locally because the processing function h is pre-defined and z_j^k is obtained by $z_j^k = z_i^k - (z_i^k - z_j^k)$, given agent i's absolute state z_i^k and its measurements $z_i^k - z_j^k \in m_i^{c,k}$ at time step k and $k+1$ respectively. The reasonable control input $u_j^{r,k}$ is given by $u_j^{r,k} = \mathcal{P}_j(z_j^k,\, I_j^{c,k}) = \mathcal{P}_i(z_i^k - (z_i^k - z_j^k),\, z_i^k - (z_i^k - I_j^{c,k}))$ where $z_i^k - I_j^{c,k} = \{z_i^k - z_{j_1}^k, \cdots, z_i^k - z_{j_p}^k\}$ and $\mathcal{N}_j^{c,k} = \{j_1, \cdots, j_p\}$. Lemma 3 ensures that $\mathcal{N}_j^{c,k} \subset \mathcal{N}_i^{f,k}$. The problem is how to determine the set $\mathcal{N}_j^{c,k}$ out of $\mathcal{N}_i^{f,k}$, corresponding to each $j \in \mathcal{N}_i^{c,k}$. Due to the fact that the control sensing range uniformly equals to r_c, $\mathcal{N}_j^{c,k}$ can be determined by validating the distance between agent j and any agent $l \in \mathcal{N}_i^f$, so that $\mathcal{N}_j^{c,k} = \{l \in \mathcal{N}_i^{f,k} | 0 < \|(z_i^k - z_j^k) - (z_i^k - z_l^k)\| < r_c\} \cup \{i\}$, where $z_i^k - z_j^k \in m_i^{c,k}$ and $z_i^k - z_l^k \in m_i^{f,k}$. Consequently agent i can predict the reasonable movement $u_j^{r,k}$ as described above and generate the residual signal r_j^k for each $j \in \mathcal{N}_i^{c,k}$, which provides the index of faults given the threshold function $\chi(|u_i^{r,k}|, \delta)$. ■

To give an example, one typical sensor model takes range and bearing measurements in 2-D state space. The above frameworks can be applied directly by replacing $z_i - z_j$ with $z_i - z_j = \begin{bmatrix} x_i - x_j \\ y_i - y_j \end{bmatrix} = \begin{bmatrix} r_{ij}\cos\theta_{ij} \\ r_{ij}\sin\theta_{ij} \end{bmatrix}$, where we denote the measurements of agent i: $r_{ij} = \|z_i - z_j\|$, $\theta_{ij} = \arctan(\frac{y_i - y_j}{x_i - x_j})$, for $j \in \mathcal{N}_i^{c,k}$. In particular, agent i can determine the network $\mathcal{N}_j^c$ as $\{l \in \mathcal{N}_i^f | 0 < \|r_{ij}^2 + r_{il}^2 - 2r_{ij}r_{il}\cos(\theta_{ij} - \theta_{il})\| < r_c\} \cup \{i\}$.

For any fault detection system, there is essentially an trade-off between the *false alarm rate* (FAR) and the *fault detection rate* (FDR) [3]. In our framework, the design preferences are embodied in the threshold function $\chi(\|u_j^r\|, \delta)$. If needed, a confirmation mechanism can be added that one agent is confirmed as faulty if it is detected with a high rate during a certain period of time, which requires more sophisticated confirmation mechanisms.

IV. Fault Isolation and Recovery

The fault detection task is usually followed by the fault isolation and recovery step to eliminate the impact of emerged faults. Regarding a multi-agent network, the task of fault isolation is simply to isolate the faulty agents so that they have no impact on the non-faulty agents. The step of fault recovery is introduced to remove the impact of faulty agents by applying external excitations [7]. Because each agent acts as a local monitor while monitored by others at the same time. There are two basic issues to be addressed: (i) when to apply the isolation or recovery step and (ii) how much the external excitation should be.

The first issue may not be trivial, particularly when we consider multi-agent systems. On the one hand, if we make the assumption that one agent $g \in V$ will be detected simultaneously by all its neighbors the moment it becomes faulty, it would be disconnected from the group simply by excluding it from all local neighboring sets, namely $\mathcal{N}_i \leftarrow \mathcal{N}_i \setminus g$, $\forall i \in V$. Thus faulty agents are isolated immediately after being detected. On the other hand, since these local monitors are running independently with different threshold functions and neighboring sets, it is more realistic to assume that one fault may be detected by different agents at different time instants. Then without the existence of a central authority, it is of vital importance that all agents synchronize the isolation and recovery step. Otherwise if one agent tries to isolate a faulty neighbor or apply extra excitation without any notice or negotiation to its neighbors, this isolation and recovery behavior will highly likely be detected as faulty, which may lead to a chain reaction in the system. To avoid this we introduce a new parameter called the fault detection and recovery cycle time $T_p = p^*T$, where the constant $p^* \in \mathbb{Z}^+$ and T is the sampling time. Typically, for every period of time T_p, each agent detects possible faults for $k \in [k^*T_p + T,\, (k^*+1)T_p - T]$ and applies the external excitation simultaneously at $k = (k^*+1)T_p,\, k^* \in \mathbb{Z}^+$.

An intuitive solution for the second issue is that after a faulty agent is detected, its accumulated contribution from $t = 0$ is removed from the whole system by applying extra excitation locally at each agent [7]. But in order to keep track of earlier contributions of faulty agents that are previously unknown, each agent needs to record the contribution of both faulty and non-faulty agents before some faults are detected. It would become infeasible especially when the underlying topology is time-varying and the running time is long. Therefore we choose to only record and remove the contribution of faulty agents after they become faulty.

Theorem 5: Suppose that the control law (2) has the summation form: $u_i^k = \mathcal{P}_i(z_i,\, I_i^{c,k}) = \sum_j p(z_i^k,\, z_j^k),\, j \in \mathcal{N}_i^{c,k}$, where $p : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}^2$. Then previously detected faults by Theorem 1 - 4 can be isolated and recovered. Moreover, the global task excluding faulty agents can be accomplished.

Proof: When the control law (2) has the summation form, we can easily separate the contribution of agent $j \in \mathcal{N}_i^{c,k}$ to agent i, i.e., $p(z_i^k, z_j^k)$. Note here $\mathcal{N}_i^{c,k}$ stands for $\mathcal{N}_i^k$ for *communication-based* models. Assume that agent g becomes faulty at time step $k = k_e > 0$ and remains faulty for $k \geq k_e$, where $k_e \in [k^*T_p{+}T,\, (k^*{+}1)T_p{-}T]$, $k^* \in \mathbb{Z}^+$. Agent i detects that agent g is faulty and starts accumulating its contribution $u_{\text{ex}} = \sum_{k\in[k_e,\,(k^*+1)T_p-T]} p(z_i^k,\, z_g^k)$ within one cycle time. Then agent i applies the external excitation $u_i = -u_{\text{ex}}$ at time step $(k^*{+}1)T_p$ and resets the accumulated value u_{ex} to zero. Fault detection mechanism is disabled at the next step $(k^*+1)T_p + T$ and resumed afterwards. Each agent repeats the same procedure independently every T_p but they share the synchronized clock. With respect to the global objective, we denote the set of faulty agents as V_f and the set of non-faulty agents as V_n so that $V_f \cup V_n = V$. If agent g is faulty for $k \geq k_e$, $g \in V_f$ for $k > k_e$. Consider the accumulated input of any agent $i \in V_n$ for $k > k_e$: $\sum_{k>k_e} u_i(t) = \sum_{k>k_e} \sum_{j\in\mathcal{N}_i} p(z_i^k,\, z_j^k) - \sum_{k>k_e} u_{\text{ex}} = \sum_{k>k_e} [\sum_{j\in\mathcal{N}_i^{c,k}} p(z_i^k,\, z_j^k) - \sum_{j\in(\mathcal{N}_i^{c,k}\cap V_f)} p(z_i^k,\, z_j^k)] = \sum_{k>k_e} \sum_{j\in(\mathcal{N}_i^{c,k}\cap V_n)} p(z_i^k,\, z_j^k)$, which means all faulty agents will not have contribution or impact on the dynamics of non-faulty agents after the time instants when they are detected as faulty. But their contribution before they are detected is still considered. Thus the final state would be the same as the system with only non-faulty agents but considering the contribution of faulty agents before they become faulty. ■

Remark: Note that the value of cycle time T_p should be relatively small to avoid abrupt changes in states when the extra excitations are applied.

V. Application to Consensus and Formation Problems

In the sequel, we apply our fault detection, isolation and recovery frameworks to two popular aspects of multi-agent cooperative control: consensus and formation problems.

A. Consensus Problem

The consensus protocol proposed in [10] for first-order systems is $u_i = \sum_{j\in\mathcal{N}_i} a_{ij}(z_j - z_i)$, where $a_{ij} > 0$ are the weights of each edge. We analyze *communication-based* and *sensing-based* models respectively.

First we assume the value of a_{ij} is identical or uniformly all one, $\forall(i,\, j) \in E$. Applying exactly the same arguments as the proofs of Theorem 1-5, we have a direct conclusion:

However it could be the case that the weights may vary with respect to different edges under different communication graphs, which means $\mathcal{P}$ is not homogeneous anymore. *Protocol I* is still useful to detect possible faults in phase one as Theorem 1 holds for both homogeneous and non-homogeneous $\mathcal{P}$. But *Protocol II* needs small modifications: instead of I_i^k, agent i transmits a new set of information $C_i^k = \{a_{ii_1}(z_{i_1} - z_i), \cdots, a_{ii_p}(z_{i_p} - z_i)\}$, where $\{i_1, \cdots, i_p\} = \mathcal{N}_i^k$, to its neighbors, so that any $j \in \mathcal{N}_i^k$ can reconstruct the reasonable input u_i^r locally.

On the other hand, for *sensing-based* models, without direct information exchange agent i needs to estimate the weights a_{jp} dynamically in real-time using only the relative state measurements $z_j - z_p$, regarding each $j \in \mathcal{N}_i^k$. But we are not going to discuss about it further here.

It is necessary to examine the final consensus value in the presence of faulty agents. Based on Theorem 5, after all faulty agents are detected they have no impact on the non-faulty agents. As the average state keeps constant under normal consensus behaviors, the final agreement point is the average state of all non-faulty agents, which is not the same as the initial average $\bar{x}(0)$ as we still consider the contribution of faulty agents before they become faulty.

B. Formation Control

Inspired by the discussion above, we consider the multi-agent formation problems, like distance-based formation [1], swarming [2] and flocking [9]. We set the desired distances d_{ij} between neighboring agents be uniformly d so that the local subjective for each agent is homogeneous.

The control protocols for distance formation [1] is $u_i = -2\sum_{j\in\mathcal{N}_i} \frac{\beta^2 - d^4}{\beta^2}(z_i - z_j)$ where $\beta = \|z_i - z_j\|^2$. The distributed control law for flocking [9] to achieve uniform distance and velocity alignment, is given by $u_i = -2\sum_{j\in\mathcal{N}_i} s_{ij}(z_j - z_i) + c(v_j - v_i)$ where c is a constant and $s_{ij} = \frac{\phi(\|z_j - z_i\| - d)}{\|z_j - z_i\|}$. $z_j - z_i$ and $v_j - v_i$ are the relative position and relative velocity for second-order systems. It is proposed for distributed swarm aggregation [2] that $u_i = -2\sum_{j\in\mathcal{N}_i} [p_{ij}(\|z_i - z_j\|) + \rho_{ij}(\|z_i - z_j\|)](z_j - z_i)$ where p_{ij} represents the attractive force and ρ_{ij} the repulsive force. Both p_{ij} and ρ_{ij} take relative states as input arguments.

Since the communication-based fault detection schemes can be easily applied using *Protocol I* and *Protocol II*. We will put emphasis on the case of *sensing-based* models.

By taking a closer look at the above listed control protocols, we may notice that the local control law is homogeneous for each agent and has the exactly the same structures.

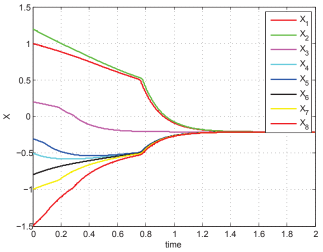

Fig. 2. Consensus without faulty agents

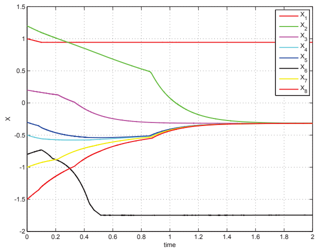

Fig. 3. One faulty agent and one malicious agent

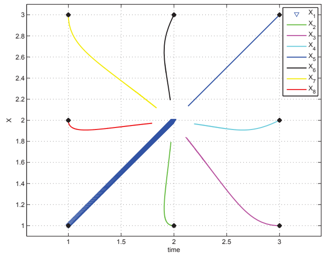

Fig. 4. Desired performance without faults

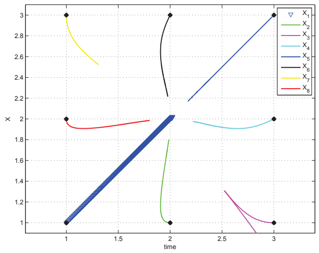

Fig. 5. One faulty agent and one malicious agent

In particular, they belong to the following generic class of control laws $u_i = \sum_{j\in\mathcal{N}_i} \rho(\|z_i - z_j\|)(z_i - z_j)$, $i \in V$ where this function $\rho(\cdot)$ only takes relative distance as input arguments. It satisfies the pre-assumption of Theorem 4 and 5 that $\mathcal{P}_i = \mathcal{P}_j$, with the summation form $\sum_{j\in\mathcal{N}_i} p(z_i, z_j)$.

VI. Simulations

We now provide computer simulations to support the presented theory. We introduce two kinds of common faults [8] in the simulation: (i) *Stuck at* and (ii) *Malicious act*.

In the first part, eight first-order agents with *sensing-based* models perform consensus in 1-D configuration space. Fig. 2 shows the desired performance without faulty behaviors, while one breaks down and another one dissipates as in Fig. 3. We investigate the case of *sensing-based* models. The underlying graph is dynamically time-varying and each agent has two neighboring set $\mathcal{N}_i^{c,k}$ and $\mathcal{N}_i^{f,k}$. Uniform weights ($a_{ij} = 1$) are chosen and $T_p = 10T$. As expected, non-faulty agents can still achieve the consensus as shown in Fig. 3, where the final agreement value depends on average state of all non-faulty agents.

The second simulation involves eight first-order agents performing distance-based formation [1] in 2-D state space. They initially form a square and the underlying topology is required to be a static star in [1]. $\mathcal{N}_1 = \{2, 3, 4, 5, 6, 7, 8\}$ and $\mathcal{N}_i = \{1\}$ for $i = 2, \cdots, 8$. Let the desired distance between neighboring agents be uniformly $d = 0.2$. Fig. 4 illustrates the formation outcome without any faulty behaviors in the group, while Fig. 5 show the scenarios where one agent breaks down and another one dissipates from the group. As expected, the distance between non-faulty neighboring agents converges to the desired value eventually.

VII. Conclusions and Future Work

This paper considers the problem of fault detection, isolation and mitigation in cooperative control of multi-agent systems. Decentralized and real-time fault detection frameworks are proposed for two different system models based on communication or relative state sensing. They are easy to implement and requiring much less computational resources. Furthermore, we address the importance of synchronization when performing fault isolation and mitigation step in multi-agent systems, where a robust solution is introduced to guarantee the global performance. Applications to the consensus and formation problems are discussed, of which the performance is verified through computer simulations.

Future research focuses on finding the fundamental limits of the presented framework, i.e., the maximal set of cooperative control laws that can be tackled with this approach. Moreover, the framework is going to be extended to more general agent dynamics.

References

[1] D.V. Dimarogonas and K.H. Johansson. Further results on the stability of distance-based multi-robot formations. *American Control Conference*, 2009.

[2] D.V. Dimarogonas and K.J. Kyriakopoulos. Connectedness preserving distributed swarm aggregation for multiple kinematic robots. *IEEE Transactions on Robotics*, pages 1213–1223, 2008.

[3] S. X. Ding. Model-based fault diagnosis techniques. *Springer*, 2008.

[4] A. Fagiolini, F. Babboni, and A. Bicchi. Dynamic distributed intrusion detection for secure multi-robot systems. *Robotics and Automation*, pages 2723 –2728, 2009.

[5] A. Fagiolini, M. Pellinacci, G. Valenti, G. Dini, and A. Bicchi. Consensus-based distributed intrusion detection for multi-robot systems. *IEEE International Conference on Robotics and Automation*, pages 120 –127, 2008.

[6] A. Fagiolini, G. Valenti, L. Pallottino, G. Dini, and A. Bicchi. Decentralized intrusion detection for secure cooperative multi-agent systems. *46th IEEE Conference on Decision and Control*, pages 1553 –1558, 2007.

[7] M. Franceschelli, M. Egerstedt, and A. Giua. Motion probes for fault detection and recovery in networked control systems. *American Control Conference*, pages 4358 –4363, june 2008.

[8] M. Franceschelli, A. Giua, and C. Seatzu. Decentralized fault diagnosis for sensor networks. *Automation Science and Engineering*, pages 334 –339, aug. 2009.

[9] R. Olfati-Saber and R.M. Murray. Flocking with obstacle avoidance: Cooperation with limited communication in mobile networks. *42st IEEE Conf. Decision and Control*, pages 2022–2028, 2003.

[10] R. Olfati-Saber and R.M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004.

[11] F. Pasqualetti, A. Bicchi, and F. Bullo. Consensus computation in unreliable networks: A system theoretic approach. *Automatic Control, IEEE Transactions on*, (99), 2011.

[12] G. Roussos and K.J. Kyriakopoulos. Completely decentralised navigation of multiple unicycle agents with prioritisation and fault tolerance. *IEEE Conference on Decision and Control*, pages 1372 –1377, 2010.

[13] I. Shames, A. M. H. Teixeira, H. Sandberg, and K. H. Johansson. Distributed fault detection for interconnected second order system. *Automatica*, Oct. 2011, to appear.

[14] S. Sundaram and C.N. Hadjicostis. Distributed function calculation via linear iterations in the presence of malicious agents, part i: Attacking the network. *American Control Conference*, june 2008.

[15] H.G. Tanner. Flocking with obstacle avoidance in switching networks of interconnected vehicles. *2004 IEEE International Conference on Robotics and Automation*, pages 3006–3011, 2004.