Monitoring Link Faults in Nonlinear Diffusively-coupled Networks

Miel Sharf, Graduate Student Member, IEEE and Daniel Zelazo, Senior Member, IEEE

Abstract—Fault detection and isolation is an area of engineering dealing with designing on-line protocols for systems that allow one to identify the existence of faults, pinpoint their exact location, and overcome them. We consider the case of multi-agent systems, where faults correspond to the disappearance of links in the underlying graph, simulating a communication failure between the corresponding agents. We study the case in which the agents and controllers are maximal equilibrium-independent passive (MEIP), and use the known connection between steady-states of these multi-agent systems and network optimization theory. We first study asymptotic methods of differentiating the faultless system from its faulty versions by studying their steady-state outputs. We explain how to apply the asymptotic differentiation to detect and isolate communication faults, with graph-theoretic guarantees on the number of faults that can be isolated, assuming the existence of a “convergence assertion protocol”, a data-driven method of asserting that a multi-agent system converges to a conjectured limit. We then construct two data-driven model-based convergence assertion protocols. We demonstrate our results by a case study.

I. INTRODUCTION

Multi-agent systems (MAS) have been widely studied in recent years, as they present both a variety of applications, as well as a deep theoretical framework [1]–[3]. One of the deepest concerns when considering applications of MAS is communication failures, which can drive the agents to act poorly, or fail their task altogether. These communication failures, which we term network failures, can either be accidental or planned by an adversary. There is a need of detecting network faults and dealing with them in real-time for the network to be secure.

Fault detection and isolation (FDI) for multi-agent systems usually deals with faults in one of the agents, see e.g. [4] and references therein. The possibility of faults in the communication links was first studied in [5] using the notion of structural controllability, which was later used in [6] to solve the problem of leader localization. The problem of network FDI, i.e., detection and isolation of network faults, was studied primarily for linear and time-invariant (LTI) systems with a known model. In [7], [8], the authors use jump discontinuities in the derivative of the output to detect topological changes in the network. Tools from switching systems theory, namely mode-observability, was used in [9] for network FDI. Combinatorial tools were used in [10], [11] to solve the FDI problem for consensus-seeking networks. Recently, [12] proposed a network FDI method which allows an uncertainty in the model, but is restricted for networks with LTI systems. A related problem in which one tries to distinguish between multi-agent systems with identical agents but different communication graphs was studied in [13]–[15], from which only [14] also deals with nonlinear agents.

We aim at a network FDI scheme applicable also for nonlinear systems by relying on another concept widespread in multi-agent systems, namely passivity. Passivity was first used to address faults by [16] for control-affine systems, although only fault-tolerance is addressed, and no synthesis procedures are suggested. Later works used FDI for a single nonlinear agent [17]–[19]. To the extent of our knowledge, passivity has not been previously used to give network FDI schemes, and no other works consider networks with nonlinear components.

Passivity theory is a cornerstone of the theoretical framework of networks of dynamical systems [20]. It allows for the analysis of multi-agent systems to be decoupled into two separate layers, the dynamic system layer and the information exchange layer. Passivity theory was first used to study the convergence properties of network systems in [21]. Many variations and extensions of passivity have been applied in different aspects of multi-agent systems. For example, the related concepts of incremental passivity or relaxed coercivity have been used to study various synchronization problems [2], [22], and more general frameworks including Port-Hamiltonian systems on graphs [23]. Passivity is also widely used in coordinated control of robotic systems [24].

One prominent variant is maximal equilibrium-independent passivity (MEIP), which was applied in [25] in order to reinterpret the analysis problem for a multi-agent system as a pair of network optimization problems. Network optimization is a branch of optimization theory dealing with optimization of functions defined over graphs [26]. The main result of [25] showed that the asymptotic behavior of these networked systems is (inverse) optimal with respect to a family of network optimization problems. In fact, the steady-state input-output signals of both the dynamical systems and the controllers comprising the networked system can be associated to the optimization variables of either an optimal flow or an optimal potential problem; these are the two canonical dual network optimization problems described in [26]. The results of [25] were used in [27], [28] in order to solve the synthesis problem for multi-agent systems, and were further used in [14], [29] to solve the network identification problem.

We aim to use the network optimization framework of [25],[27],[28] for analysis and synthesis of multi-agent systems in order to provide a strategy for detecting and isolating network faults. We also consider adversarial games regarding communication faults. We strive to give graph-theoretic-based results, showing that network fault detection and isolation
can be done for any MEIP multi-agent system, so long that the graph \( G \) satisfies certain conditions. We show that if the graph \( G \) is “connected enough”, we can solve the network FDI problem. Namely, if \( G \) is 2-connected, then detecting the existence of any number of faults is possible, and if \( G \) is \( k \)-connected with \( k > 2 \), we can isolate up to \( k - 2 \) faults.

The rest of the paper is as follows. Section II surveys the relevant parts of the network optimization framework. Section III presents the problem formulation of this work and states the assumptions used throughout the paper. Section IV presents notations: We use basic notions from algebraic graph theory \([30]\). An undirected graph \( G = (\mathbf{V}, \mathbf{E}) \) consists of a finite set of vertices \( \mathbf{V} \) and edges \( \mathbf{E} \subset \mathbf{V} \times \mathbf{V} \). We denote by \( e = \{i, j\} \in \mathbf{E} \) the edge that has ends \( i \) and \( j \) in \( \mathbf{V} \). For each edge \( e \), we pick an arbitrary orientation and denote \( e = (i, j) \). The incidence matrix of \( G \), denoted \( \mathbf{E}_G \in \mathbb{R}^{|\mathbf{E}| \times |\mathbf{V}|} \), is defined such that for edge \( e = (i, j) \in \mathbf{E} \), \( [\mathbf{E}_G]_{ie} = +1 \), \( [\mathbf{E}_G]_{je} = -1 \), and \( [\mathbf{E}_G]_{ik} = 0 \) for \( k \neq i, j \). We also use simple notions from graph theory \([31]\). A path is a sequence of distinct nodes \( v_1, v_2, \ldots, v_n \) such that \( \{v_i, v_{i+1}\} \in \mathbf{E} \) for all \( i \). A cycle is the union of a path \( v_1, \ldots, v_n \) with the edge \( \{v_1, v_n\} \). A cycle \( v_1, v_2, \ldots, v_n-1, v_1 \) is called simple if \( v_i \neq v_j \) for all \( i \neq j \). A collection of paths is called vertex-disjoint if no two share a node, except possibly for their first and last nodes. Furthermore, for a linear map \( T : U \to V \) between vector spaces, we denote the kernel of \( T \) by \( \ker T \).

II. NETWORK OPTIMIZATION AND MEIP MULTI-AGENT SYSTEMS

The role of network optimization theory in cooperative control was introduced in \([25]\), and was used in \([27], [28]\) to solve the synthesis problem for MAS. In this section, we briefly on the main results we need from \([25], [27], [28]\).

A. Diffusively-Coupled Networks and Their Steady-States

Consider a collection of SISO agents interacting over a network \( G = (\mathbf{V}, \mathbf{E}) \). The agents \( \{\Sigma_i\}_{i \in \mathbf{V}} \) and the controllers \( \{\Pi_e\}_{e \in \mathbf{E}} \) have the following models:

\[
\begin{align*}
\Sigma_i : & \begin{cases}
\dot{x}_i = f_i(x_i, u_i) \\
y_i = h_i(x_i)
\end{cases}, \quad \Pi_e : \begin{cases}
\dot{\eta}_e = \phi_e(\eta_e, \zeta_e) \\
\mu_e = \psi_e(\eta_e, \zeta_e)
\end{cases}
\end{align*}
\]

We consider stacked vectors of the form \( u = [u_1^\top, \ldots, u_n^\top]^\top \) and similarly for \( x, y, \zeta, \eta \) and \( \mu \). The agents and controllers are coupled by defining the controller input as \( \zeta = \mathbf{E}_G^\top \gamma \) and the control input as \( u = -\mathbf{E}_G \mu \). This closed-loop system is called a diffusively-coupled network, and is denoted by \((G, \Sigma, \Pi)\). Its structure is illustrated in Figure 1. We wish to study the steady-states of the closed-loop. Suppose that \((u, y, \zeta, \mu)\) is a steady-state of \((G, \Sigma, \Pi)\). For every \( i \in \mathbf{V}, e \in \mathbf{E} \), \((u_i, y_i)\) is a steady-state input-output pair of the \( i \)-th agent, and \((\zeta_e, \mu_e)\) is a steady-state pair of the \( e \)-th controller. This motivates the following definition, originally from \([25]\):

Definition 1. The steady-state input-output relation \( k \) of a dynamical system is the collection of all steady-state input-output pairs of the system. Given a steady-state input \( u \) and a steady-state output \( y \), we define

\[
k(u) = \{y : (u, y) \in k\}, \quad k^{-1}(y) = \{u : (u, y) \in k\}.
\]

Let \( k_i \) be the steady-state input-output relation for the \( i \)-th agent, \( g_e \) be the steady-state input-output relation for the \( e \)-th controller, and \( k, g \) be their stacked versions. Then the closed-loop steady-state \((u, y, \zeta, \mu)\) has to satisfy \( y \in k(u), \zeta = \mathbf{E}_G^\top y, \mu = g(\zeta), u = -\mathbf{E}_G \mu \). By a simple manipulation, one can show that \( y \) is a closed-loop steady-state for the agent output if and only if \( 0 \in k^{-1}(y) + \mathbf{E}_G^\top g(\mathbf{E}_G^\top y) \) \([28]\).

B. MEIP Systems and Closed-Loop Convergence

The convergence of the diffusively-coupled network \((G, \Sigma, \Pi)\) can be assured using passivity. We first recall the classic definition of (shifted) passivity:

Definition 2 (Passivity \([22]\)). Let \( \Upsilon \) be a SISO system with input \( u(t) \), output \( y(t) \) and state \( x(t) \), and let \((u, y)\) be a steady-state input-output pair of the system. For a differentiable function \( S = S(x) \) and a number \( \rho > 0 \), we define the inequality \( \frac{d}{dt} S(x) \leq -\rho \|y(t) - y\|^2 + (y(t) - y)(u(t) - u) \) We say \( \Upsilon \) is passive (w.r.t. \((u, y)\)) if there exists a semi-definite storage function \( S(x) \) and \( \rho \geq 0 \) such that the inequality holds for any trajectory. Also, we say the system is output-strictly passive (w.r.t. \((u, y)\)) if the same condition holds for some \( \rho > 0 \). The largest number \( \rho \) for which the condition holds is called the (output) passivity index w.r.t. \((u, y)\).

Passivity was first used for diffusively-coupled networks in \([21]\). It is known that if \((u, y, \zeta, \mu)\) is an equilibrium of the network, and the agents and controllers are passive with respect to \((u_i, y_i)\) and \((\zeta_e, \mu_e)\), then the network converges to said equilibrium. The existence of an equilibrium for the closed-loop network can be proved using network optimization tools under certain monotonicity assumptions on the steady-state input-output relation of the agents and controllers \([25], [27], [28]\), namely under the following variant of passivity:

Definition 3 (Maximal Equilibrium Independent Passivity \([25]\)). Consider the SISO dynamical system of the form

\[
\Upsilon : \dot{x} = f(x, u); \quad y = h(x, u),
\]
with input-output relation \( r \). The system \( \Sigma \) is said to be (output-strictly) MEIP if the following conditions hold:

i) The system \( \Sigma \) is (output-strictly) passive with respect to any steady-state pair \((u, y)\), i.e., with respect to any \( u, y \) such that \( y \in r(u) \).

ii) The steady-state input-output relation \( r \) is maximally monotone. That is, if \( (u_1, y_1), (u_2, y_2) \in r \) then \( u_1 - u_2, y_1 - y_2 \geq 0 \), and \( r \) is not contained in any larger monotone relation [26].

The passivity index of the system \( \Sigma \) is defined as \( \min_{y \in r(u)} \rho_{u,y} \), where \( \rho_{u,y} \) is the passivity index with respect to \((u, y)\).

Such systems include single integrators, gradient systems, port-Hamiltonian systems on graphs, and others (see [25], [28] for more examples). In this work we often consider networks of control-affine systems. The theorem below gives a sufficient condition for a control-affine system to be MEIP:

**Theorem 1.** Let \( \Sigma \) be the SISO system of the form \( \dot{x} = -f(x) + q(x)u, y = h(x) \). Suppose that \( q(x) \) is positive for all \( x \), that \( h \) is strictly monotone \( C^1 \) ascending, and that \( f/q \) is \( C^1 \) and monotone ascending.

1) A pair \((u, \gamma)\) is a state-space input-output pair for \( \Sigma \) if and only if there exists some \( x \in \mathbb{R} \) such that \( u = f(x)/q(x) \) and \( y = h(x) \).

2) For any \( x \in \mathbb{R} \), the function \( S(x) = \int_x^\infty h(\sigma) - h(x) \frac{d\sigma}{q(\sigma)} \) is a storage function for the state-space input-output pair \( u = f(x)/q(x) \) and \( y = h(x) \).

3) The function \( S(x) \) proves that \( \Sigma \) is passive w.r.t. \((u, y)\) with passivity index \( \rho = \inf_{x \in \mathbb{R}} \frac{S(x)}{h(x) - h(x)} \geq 0 \).

4) If either \( \lim_{t \to \infty} |f(t)|/|q(t)| = \infty \) or \( \lim_{t \to \infty} |h(t)| = \infty \), then the system is MEIP.

5) If the derivative of \( h \) is always positive, then the inverse steady-state relation \( k^{-1} \) is differentiable.

**Proof.** The first, second and fourth parts are proved in [33, Proposition 1]. As for the third, we note that:

\[
\dot{S} = \left( \frac{h(x) - h(x)}{q(x)} \right) \dot{x} = \left( \frac{h(x) - h(x)}{q(x)} \right) (-f(x) + q(x)u) = (h(x) - y)(u - u) - (h(x) - h(x)) \left( \frac{f(x)}{q(x)} - \frac{f(x)}{q(x)} \right) \\
\leq (y - y)(u - u) - \rho(y - y)^2
\]

We note that \( \rho \geq 0 \) as \( h(x) - h(x) \left( \frac{f(x)}{q(x)} - \frac{f(x)}{q(x)} \right) \geq 0 \) by monotonicity. Moreover, we note that \( \int_x^\infty h(\sigma) - h(x) \frac{d\sigma}{q(\sigma)} \geq 0 \), with strict inequality whenever \( x \neq x \), as \( h \) is strictly monotone and \( q(x) > 0 \). Thus \( S \) is a \( C^1 \) storage function, and we conclude the system is passive with passivity index \( \rho \geq 0 \) with respect to the steady-state input-output pair \((u, y)\). Lastly, we note that because \( h \) is strictly monotone, the inverse \( h^{-1} \) can be defined. Thus, the inverse steady-state relation \( k^{-1} \) is given by \( k^{-1}(y) = \left( \frac{f(h^{-1}(y))}{q(h^{-1}(y))} \right) \), which is differentiable by the inverse function theorem.

As we previously remarked, MEIP can be used to prove existence of a closed-loop equilibrium for networks:

**Theorem 2 ([25], [27]).** Consider the network \((\mathcal{G}, \Sigma, \Pi)\). Assume the agents \( \Sigma_i \) are MEIP, and the controllers \( \Pi_e \) are output-strictly MEIP (or vice versa). Then the signals \( u(t), y(t), \zeta(t), \mu(t) \) of the closed-loop system converge to steady-state values \( u, y, \zeta, \mu \) where \( 0 \leq k^{-1}(y) + \mathcal{E} \gamma_q(\hat{E}_q y) \).

**C. The Synthesis Problem for MEIP Multi-Agent Systems**

The synthesis problem of MAS with MEIP agents has been studied in [27], [28]. The problem deals with synthesizing controllers \( \{\Pi_e\} \) forcing the closed-loop network to converge to some desired steady-state output \( y^\ast \), when the agents \( \Sigma \) and the graph \( \mathcal{G} \) are known. We cite the following results from [28]:

**Theorem 3 ([28]).** Let \( \Sigma \) be any MEIP agents and let \( \mathcal{G} \) be any graph. Let \( y^\ast \in \mathbb{R}^{|\mathcal{V}|} \) be any desired steady-state output. Then there exists a solution to the synthesis problem (i.e., a realization of the controllers \( \Pi \) with desired output \( y^\ast \) for which the controllers are output-strictly MEIP.

**Remark 1.** The paper [28] depicts many possible solutions to the synthesis problem with output-strictly MEIP controllers. It is shown that one can always solve the problem using affine controllers. Another suggested solution is an augmentation of any preferred output-strictly MEIP controller with a constant exogenous input. In practice, we will usually opt for the augmentation procedure when using the theorem as a tool for synthesis, as many real-world networks are already equipped with some given edge controllers. If this is not the case, one can use the affine controllers instead.

### III. Problem Formulation

This section presents the problem we aim to solve, and states the assumptions we make to tackle it. We consider a diffusively-coupled network of the form \( \mathcal{G} = (\mathcal{G}, \Sigma_i) \subseteq \mathcal{V} \), where \( \mathcal{G} = (\mathcal{V}, \mathcal{E}_G) \) is the interaction graph, \( \Sigma_i \) are the agents, and \( \mathcal{E}_G \) are the edge controllers. For any subgraph \( \mathcal{H} = (\mathcal{V}, \mathcal{E}_{\mathcal{H}}) \) of \( \mathcal{G} \), we can consider another diffusively-coupled network \( \mathcal{G}_{\mathcal{H}} = (\mathcal{G}, \Sigma_i) \subseteq \mathcal{V} \), where \( \mathcal{E}_{\mathcal{H}} \) corresponds to the edges \( \mathcal{E} \setminus \mathcal{E}_G \) have malfunctioned and stopped working. Edges can fault mid-run, but we assume that once an edge has malfunctioned, it remains faulty for the remainder of the run. If we let \( \mathcal{G} \) be the collection of all nonempty subgraphs of \( \mathcal{G} \), then one can think of the closed-loop diffusively-coupled network as a switched system, where the switching signal \( \varsigma : [0, \infty) \to \mathcal{G} \) designates the functioning edges at each time instant. The assumption that faulty edges remain faulty throughout the run can be described using the switching signal \( \varsigma \). Namely, we require that the switching signal \( \varsigma \) is non-increasing, in the sense that for all times \( t_1 < t_2 \), \( \varsigma(t_2) \) is a subgraph of \( \varsigma(t_1) \). We denote the number of faulty edges at time \( t \) by \( \varsigma(t) \).

Now, consider a collection of agents \( \{\Sigma_i\} \) and a graph \( \mathcal{G} \). Fix some constant vector \( y^\ast \in \mathbb{R}^{|\mathcal{V}|} \). Our goal is to design a control scheme for which the closed-loop network will converge to the steady-state output \( y^\ast \). In the absence of faults, we can solve the synthesis problem as in Theorem 3. However, designing controllers while ignoring faults might prevent the system from achieving the control goal. For that reason, we...
also seek for a fault monitoring system, consisting of the agents and networked controllers, that attempts to identify aberrant behavior. When it does, it declares a fault [34]. We now formulate the problems of network fault detection and isolation:

**Problem 1** (Network Fault Detection). Let \( \{\Sigma_i\}_{i \in \mathcal{V}} \) be a set of agents, \( \mathcal{G} \) be a graph, \( y^* \) be any desired steady-state output, and let \( \varsigma(t) \) be any non-increasing switching signal. Find edge controllers \( \{\Pi_e\}_{e \in \mathcal{E}_G} \) and a monitoring system such that,

i) if no faults occur, i.e. \( \varsigma(t) = \mathcal{G}, \forall t \), then the closed-loop diffusively-coupled network converges to the steady-state output \( y^* \), and the fault monitoring system never declares a fault;

ii) if faults do occur, i.e. \( \exists t, \varsigma(t) \neq \mathcal{G} \), then the fault monitoring system declares a fault.

**Problem 2** (Network Fault Detection and Isolation). Let \( \{\Sigma_i\}_{i \in \mathcal{V}} \) be a set of agents, \( \mathcal{G} \) be a graph, and \( y^* \) be any desired steady-state output. Given some \( r < |\mathcal{E}_G| \), find a synthesis for the edge controllers such that for any monotone \( r \), \( \forall t \), the closed-loop diffusively-coupled network converges to the steady-state output \( y^* \), i.e., the effect of up to \( r \) faults can be isolated from the network\(^1\).

### A. Assumptions

We now state the assumptions used throughout the work. For the remainder of this work, we fix the agents \( \{\Sigma_i\} \), and make the following assumption.

**Assumption 1.** The agent dynamics \( \{\Sigma_i\} \) are MEIP, and the chosen controller dynamics \( \{\Pi_e\} \) are output-strictly MEIP (or vice versa). Moreover, the relations \( k_{i}^{-1}, \gamma_{e} \) are \( C^1 \) functions. Furthermore, the derivative \( \frac{dk_{i}^{-1}}{dy_{e}} \) is positive at any \( y_{e} \in \mathbb{R} \).

The passivity assumption assures that all the systems \( N^{\mathcal{H}}_{i} \) will globally asymptotically converge to some limit. The added smoothness assumptions, together with the positive derivative assumption, are technical assumptions that are needed to apply tools from manifold theory that will be used later. The passivity assumption allows the consideration of, e.g., port-Hamiltonian systems and gradient-descent systems [25]. Moreover, if a system satisfies any dissipation inequality with respect to all equilibria, one can use output-feedback and input-feedthrough to force MEIP [35]. Theorem 1 shows that the smoothness assumption holds for many control-affine systems. Moreover, it can be easily shown using the definition of passivity that if \( \Sigma_{t} \) is output-strictly MEIP with passivity index \( \rho \), then \( \frac{dk_{i}^{-1}}{dy_{e}} \geq \rho > 0 \) whenever \( k_{i}^{-1} \) is differentiable. Furthermore, the smoothness assumption can be relaxed by allowing \( k_{i}^{-1}, \gamma_{e} \) to not be differentiable at finitely many points. The arguments presented below still hold, but require heavier tools from measure theory, so we avoid them for clarity of presentation.

In some cases, we need to sense the state of the system, including the state of the controllers. Sometimes, the control model is such that the controller state has a physical meaning that can be measured even for non-connected agents. For example, in the traffic control model in [36], the state \( \eta_{i} \) is the relative position between two vehicles. However, the controller state of some systems might not have a physical meaning. For example, consider a collection of robots trying to synchronize their positions, where the output \( y(t) \) is the position of each robot and the edge controllers are PI controllers. In that case, the controller state \( \eta(t) \) has no physical meaning, and thus cannot be defined for non-connected agents. Some of the techniques developed later require us to be able to sense the state of the system, including the controllers’ states. Thus, we will sometimes make the following assumption:

**Assumption 2.** The controllers \( \Pi_{e} \) are static nonlinearities given by the functions \( g_{e} \), i.e., \( \mu_{e}(t) = g_{e}(\varsigma_{e}(t)) \) for all \( t \). In this case, the steady-state relation \( \gamma_{e} \) is equal to the function \( g_{e} \), and the closed-loop system is \( \dot{x} = f(x, -E_{G}g(E_{G}h(x))) \), or equivalently, \( \dot{x}_{i} = f_{i} \left( x_{i}, \sum_{e \in \mathcal{E}(i,j)} g_{e}(h_{j}(x_{j}) - h_{i}(x_{i})) \right) \).

In one of the methods below, we will want to have a clear relationship between the measurements \( h_{i}(x_{i}) \) and the storage functions \( S_{i}(x_{i}) \). To achieve this, we follow Theorem 1 and assume that the agents are control-affine:

**Assumption 3.** Assumption 2 holds, and the agents have the form \( \dot{x}_{i} = -f_{i}(x_{i}) + q_{i}(x_{i})u_{i} \), \( y_{i} = h_{i}(x_{i}) \). Thus, the closed-loop system is governed by:

\[
\dot{x}_{i} = -f_{i}(x_{i}) + g_{i}(x_{i}) \sum_{e \in \mathcal{E}(i,j)} g_{e}(h_{j}(x_{j}) - h_{i}(x_{i})). \tag{3}
\]

It should be noted that the MEIP property for the static controllers \( g_{e} \) reduces to monotonicity of the functions \( g_{e} \).

In the next section, we start heading toward a solution to Problems 1 and 2. We do so by exhibiting a method for asymptotically differentiating between the nominal dynamical system \( N^{\mathcal{H}}_{G} \) and the faulty dynamical systems \( N^{\mathcal{H}}_{t} \). Later, we show how this asymptotic differentiation can induce a finite-time differentiation of the systems.

### IV. ASYMPTOTIC DIFFERENTIATION BETWEEN NETWORKS

In this section, we develop the notion of edge-indication vectors, which will be used for network fault detection later. In [14], the notion of indication vectors was first developed. These are constant exogenous inputs used to drive the closed-loop system, chosen appropriately to give different steady-state limits for systems with identical agents and controllers, but different underlying graphs. The idea of using constant exogenous inputs to drive the network into favorable steady-state outputs was also used in [29] to give a network reconstruction algorithm with optimal time complexity, although it considers sets of multiple constant exogenous inputs applied in succession. Here, we opt for a slightly different strategy.

In [14], [29], the problem of network reconstruction was considered, in which we cannot affect the agents, controllers, or the underlying graph. In network FDI, we are doing synthesis, so we can manipulate the controllers and (in most cases) the underlying network. For that reason, we opt for a

---

1 Some authors refer to fault isolation in this case as identifying the faulty links, which is achieved by the algorithms in Subsection V.C as a side effect.
slightly different idea, in which we add a constant exogenous signal to the output of the controllers, that is, we consider \( u(t) = -E_G(\mu(t) + \omega) \). A system implementing this control law is said to have the interaction protocol \((\Pi, w)\). Analogously to the notion of indication vectors, we desire that networks with identical agents and controllers, but different underlying graphs, will be forced to converge to different steady-state outputs. This is because we can monitor the output \( y \) of the system and use it to detect changes in the underlying graph, i.e., network faults. For that, we first determine what the steady-state limit is for these systems \((G, \Sigma, (\Pi, w))\).

**Proposition 1.** Consider a diffusively-coupled network \( N_H = (H, \Sigma, \Pi) \) satisfying Assumption 1. Suppose that \( w \in \mathbb{R}^{[E_H]} \) is any constant signal added to the controller output, i.e., the loop is closed as \( u(t) = -E_H(\mu(t) + \omega) \). Then \( y \) is a steady-state output of the closed-loop system if and only if

\[
k^{-1}(y) + E_H \gamma(E_H^T y) = -E_H w.
\]

**Proof.** Follows from the discussion after Definition 1 as the new steady-state relation for the controllers is given as \( \gamma(\zeta) = \gamma(\zeta) + \omega \).

In our case, the constant signal \( w \) will be in \( \mathbb{R}^{[E_H]} \), as we determine the exogenous controller output on each edge of \( G \). If one then considers the system \( N_H \) for some \( H \in \mathcal{G} \), then the exogenous controller output will be different from \( w \), as it will only have entries of \( w \) corresponding to edges in \( H \). To formulate this, take any graph \( H \in \mathcal{G} \), and let \( P_H \) be the linear map \( \mathbb{R}^{[E_H]} \to \mathbb{R}^{[E_H]} \) removing entries corresponding to edges absent from \( H \). In other words, this is a \( \mathbb{R}^{[E_H] \times [E_H]} \) matrix with entries in \( \{0, 1\} \), whose rows are the rows of the identity matrix \( I_d \in \mathbb{R}^{[E_H] \times [E_H]} \) corresponding to the edges of \( H \).

We can now define the notion of edge-indication vectors.

**Definition 4.** Let \((G, \Sigma, \Pi)\) be a network satisfying Assumption 1. Let \( w \in \mathbb{R}^{[E_G]} \) be any vector, and for any graph \( H \in \mathcal{G} \), we denote the solution of (4) with underlying graph \( H \) and exogenous input \( P_H w \) by \( y_H \).

- The vector \( w \) is called a \((\mathcal{G}, H)\)-edge-indication vector if for any \( H' \in \mathcal{G} \) such that \( H' \neq H \), we have \( y_{H'} \neq y_H \).
- The vector \( w \) is called a \( \mathcal{G} \)-edge-indication vector if for any two graphs \( H_1 \neq H_2 \) in \( \mathcal{G} \), \( y_{H_1} \neq y_{H_2} \).

**Note 1.** An edge-indication vector is a bias chosen on each edge in \( G \). This bias can be programmed into the controllers and nodes, and need not be changed nor computed on-line. In this light, for any \( w \in \mathbb{R}^{[E_G]} \), (4) transforms into

\[
k^{-1}(y) + E_H \gamma(E_H^T y) = -E_H P_H w,
\]

We wish to find a \( \mathcal{G} \)-edge-indication vector for given agents and controllers, or at least a \( \mathcal{G}, G \)-edge-indication vector. As in [14], we use randomization. We claim that random vectors are \( \mathcal{G} \)-edge-indication vectors with probability 1.

**Theorem 4.** Let \( \mathbb{P} \) be an absolutely continuous\(^3\) probability measure on \( \mathbb{R}^{[E_G]} \). Let \( w \) be a vector sampled according to \( \mathbb{P} \). Then \( \mathbb{P}(w \text{ is a } \mathcal{G} \text{-edge-indication vector}) = 1 \).

\(^3\)Unless stated otherwise, absolute continuity is with respect to the Lebesgue measure.

**Proof.** From the definition, \( w \) is not a \( \mathcal{G} \)-edge-indication vector if and only if there are two graphs \( G_1, G_2 \in \mathcal{G} \) such that the same vector \( y \) solves equation (5) for both graphs. We show that for any \( G_1, G_2 \in \mathcal{G} \), the probability that the two equations share a solution is zero.

Let \( n \) be the number of vertices in \( G \). For each graph \( H \in \mathcal{G} \), define a function \( F_H : \mathbb{R}^n \times \mathbb{R}^{[E_G]} \to \mathbb{R}^n \) by \( F_H(y, w) = k^{-1}(y) + E_H \gamma(E_H^T y) + E_H P_H w \). The set of steady-state exogenous input and output pairs for the system \( N_H \) is given by the set \( A_H = \{ (y, w) : F_H(y, w) = 0 \} \). We note that the differential \( dF_H \) always has rank \( n \). Indeed, it can be written as \( [\nabla k^{-1}(y) + E_H \nabla \gamma(E_H^T y), E_H P_H] \), where \( \nabla \gamma(E_H^T y) \in \mathbb{R}^{[E_H] \times n} \). By assumption 1, the first matrix, of size \( n \times n \), is positive-definite as a sum of a positive-definite matrix and a positive semi-definite matrix, hence invertible. Thus, by the implicit function theorem, \( A_H \) is a manifold of dimension \( |E_G| \). Moreover, by Assumption 1 for any \( w \) there is a unique \( y \) such that (4) is satisfied. Thus, \( \mathbb{P} \) gives rise to an absolutely continuous\(^4\) probability measure on each manifold \( A_H \). Hence, it is enough to show that for any \( G_1 \neq G_2 \), the intersection \( A_{G_1} \cap A_{G_2} \) has dimension \( \leq |E_G| - 1 \).

To show this, we take any point \( (y, w) \in A_{G_1} \cap A_{G_2} \). As both \( A_{G_1}, A_{G_2} \) are of dimension \( |E_G| \), it is enough to show that they do not have the same tangent space at \( (y, w) \). The tangent space of the manifold \( A_H \) is given by the kernel of the differential \( dF_H(y, w) : \mathbb{R}^n \times \mathbb{R}^{[E_G]} \to \mathbb{R}^n \), so we show that if \( G_1 \neq G_2 \), the kernels \( \ker dF_{G_1}, \ker dF_{G_2} \) are different at \( (y, w) \). As \( G_1 \neq G_2 \), we can find an edge existing in one of the graphs and not the other. Assume without loss of generality that the edge \( e \) exists in \( G_1 \) but not in \( G_2 \), and let \( v = (0, I_e) \), where \( I_e \) is the vector in \( \mathbb{R}^{[E_H]} \) with all entries zero, except for the \( e \)-th entry, which is equal to 1. Then \( v \in \ker dF_{G_1} \) and only if \( I_e \in \ker(E_H P_{G_1}) \). It is clear that \( I_e \notin \ker(E_H P_{G_1}) \), as \( P_{G_1} I_e = I_e \), and thus \( E_{G_1} P_{G_1} I_e = E_{G_1} I_e \neq 0 \). Moreover, \( I_e \in \ker(E_{G_2} P_{G_2}) \), as \( P_{G_2} I_e = 0 \), so \( \ker dF_{G_1} \neq \ker dF_{G_2} \) at \( (y, w) \). Thus \( A_{G_1} \cap A_{G_2} \) is of dimension \( \leq |E_G| - 1 \), meaning that it is a zero-measure set inside both \( A_{G_1}, A_{G_2} \).

**Theorem 4** presents a way to choose a \( \mathcal{G} \)-edge-indication vector, but does not deal with the control goal. One could satisfy the control goal by using Theorem 4 to solve the synthesis problem for the original graph \( G \), but we cannot assure we get an edge-indication vector. Note that any \( w \in \ker E_G P_{G_2} \) gives a solution of (5) identical to the solution for \( w = 0 \). Thus, choosing an exogenous control input in \( \ker E_G P_{G_2} \) does not change the steady-state output of the system \( N_G \). However, it does change the steady-state output of all other systems \( N_H \). This suggests to search for an edge-indication vector in \( \ker E_G P_{G_2} \). We show that this is possible if \( G \) is “sufficiently connected”, defined below in an exact manner.

**Proposition 2** (Menger’s Theorem [31]). Let \( G \) be any connected graph. The following conditions are equivalent:

1. Between every two nodes there are \( k \) vertex-disjoint simple paths.

\(^4\)With respect to the \( |E_G| \)-dimensional Hausdorff measure, or equivalently, with respect to the standard Riemannian volume form on \( A_H \).

\(^5\)As the push-forward measure of \( \mathbb{P} \) under the map \( w \mapsto (\varphi(w), w) \), where \( \varphi \) is the local map \( w \mapsto y \) given by the implicit function theorem.
2) For any $k - 1$ vertices $v_1, \ldots, v_{k-1} \in V$, the graph $G - \{v_1, \ldots, v_{k-1}\}$ is connected.

Graphs satisfying either of these conditions are called $k$-connected graphs.

We will take special interest in 2-connected graphs. Specifically, we can state the following theorem about edge-indication vectors in $\ker E_G P_G$.

**Theorem 5.** Let $P$ be any absolutely continuous probability distribution on $\ker E_H P_H$, where $H$ is 2-connected. Suppose furthermore that $w$ is a vector sampled according to $P$. Then $P(w) = a(\Theta, H)$-edge-indication vector) $= 1$.

We first need to state and prove a lemma:

**Lemma 1.** Let $H$ be a 2-connected graph. Suppose we color the edges of $H$ in two colors, red and blue. If not all edges have the same color, then there is a simple cycle in $H$ with both red and blue edges.

**Proof.** Suppose, heading toward contradiction, that any simple cycle in $H$ is monochromatic. We claim that for each vertex $x$, all the edges touching $x$ have the same color. Indeed, take any vertex $x$, and suppose that there are two neighbors $v_1, v_2$ of $x$ such that the edge $\{x, v_1\}$ is blue and the edge $\{x, v_2\}$ is red. We note that $v_1 \rightarrow x \rightarrow v_2$ is a path from $v_1$ to $v_2$, meaning there is another path from $v_1$ to $v_2$ which does not pass through $x$. Adding both edges to the path yields a simple cycle with edges of both colors, as $\{x, v_1\}$ is blue and $\{x, v_2\}$ is red. Thus, every node touches edges of a single color.

Let $\mathcal{V}_{red}$ be the set of nodes touching red edges, and $\mathcal{V}_{blue}$ be the set of nodes touching blue edges. We know that $\mathcal{V}_{red}$ and $\mathcal{V}_{blue}$ do not intersect. Moreover, if we had an edge between $\mathcal{V}_{red}$ and $\mathcal{V}_{blue}$, it had a color. Assume, without loss of generality, it is blue. That would mean some vertex in $\mathcal{V}_{red}$ would touch a blue edge, which is impossible. Thus there are no edges between $\mathcal{V}_{red}$ and $\mathcal{V}_{blue}$. By assumption, there is at least one edge of each color in the graph, meaning that both sets are nonempty. Thus we decomposed the set of vertices in $H$ to two disjoint, disconnected set. As $H$ is a connected graph, we find a contradiction and complete the proof. □

We can now prove Theorem 5.

**Proof.** We denote $m_1 = \dim \ker E_H P_H$. The proof is similar to the proof of Theorem 4. We again define functions $F_{\tilde{G}_1}$ for graphs $G_1 \in \mathcal{G}_n$ as $F_{\tilde{G}_1}(y, w) = k^{-1}(y) + E_{\tilde{G}_1} g(E_{\tilde{G}_1} y) + E_{\tilde{G}_1} F_{\tilde{G}_1} w$, but this time we consider the function $F_{\tilde{G}_1}$ as defined on the space $\ker E_H P_H \subset \mathbb{R}^{dF_H}$. As before, we define $A_{\tilde{G}_1} = \{y, w : F_{\tilde{G}_1}(y, w) = 0\}$ and use the implicit function theorem to show that $A_{\tilde{G}_1}$ are all manifolds, but their dimension this time is $m_1 = \dim \ker E_H P_H$. This time, we want to show that if $H \neq G_1$, then $A_{\tilde{G}_1} \cap A_{\tilde{G}_2}$ is an embedded sub-manifold of dimension $\leq m_1 - 1$, as we want to show that $(\gamma, w)$, the solutions with graph $G_1$ and graph $H$ are different. As before, it is enough to show that if $(\gamma, w) \in A_{\tilde{G}_1} \cap A_{\tilde{G}_2}$ then the kernels $\ker dF_{\tilde{G}_1}$ and $\ker dF_{\tilde{H}}$ are different at $(\gamma, w)$. We compute that for any graph $G_1$,

$$dF_{\tilde{G}_1} = [\nabla k^{-1}(y) + E_{\tilde{G}_1} \gamma(\dot{E}_{\tilde{G}_1}^T y), (E_{\tilde{G}_1} P_{\tilde{G}_1})|_{\ker E_H P_H}]$$

where $[\ker E_H P_H]$ is the restriction of the matrix to $\ker E_H P_H$.

Thus, if $G_1$ is any graph in $\mathcal{G}$ which is not a subgraph of $H$, it contains an edge $e$ absent from $H$. Following the proof of Theorem 4 word-by-word, noting that $\mathbb{I}_e \in \ker E_H P_H$, we conclude that the ker $dF_{\tilde{G}_1}$ and ker $dF_{\tilde{H}}$ are different at $(y, w)$. Thus we restrict ourselves to non-empty subgraphs $G_1$ of $H$.

For any collection $E$ of edges in $\ker H$, we consider $v = (0, \mathbb{I}_E)$, where $\mathbb{I}_E$ is equal to $\sum_{e \in E} \mathbb{I}_e$. If $E$ is a set of edges of a cycle in $H$, then the vector $v$ lies in the kernel of $dF_H$. We show that there is some cycle in $H$ such that $v$ does not lie in the kernel of $dF_{\tilde{G}_1}$, completing the proof.

The graph $G_1$ defines a coloring of the graph $H$ - edges in $G_1$ are colored in blue, whereas edges absent from $G_1$ are colored in red. Because $G_1$ is a non-empty proper subgraph of $H$, this coloring contains both red and blue edges. By the lemma, there is a simple cycle in $H$ having both red and blue edges. Let $E$ be the set of the edges traversed by the cycle. We claim that $E_{\tilde{G}_1} P_{\tilde{G}_1} \mathbb{I}_E \neq 0$, which will complete the proof of the theorem. Indeed, because the simple cycle contains both red and blue edges, we can find a vertex touching both a red edge in the cycle and a blue edge in the cycle. We let $v$ be the vertex, and let $e_1, e_2$ be the corresponding blue and red edges. Recalling the cycle is simple, these are the only cycle edges touching $v$. However, by the coloring, $e_1$ is in $G_1$, but $e_2$ is not. Thus,

$$(E_{\tilde{G}_1} P_{\tilde{G}_1} \mathbb{I}_E)_v = (E_{\tilde{G}_1} P_{\tilde{G}_1} e_1, e_1) + (E_{\tilde{G}_1} P_{\tilde{G}_1} e_2, e_2) = (E_{\tilde{G}_1} e_1, e_1) = \pm 1 \neq 0,$$

and in particular, $\mathbb{I}_E \notin \ker E_{\tilde{G}_1} P_{\tilde{G}_1}$. □

**V. Monitoring Network Faults**

In this section, we consider two applications of the developed framework, namely network fault detection and isolation, and defense strategies for adversarial games over networks. We first present a simple algorithm for network fault detection. Then, we discuss defense strategies for adversarial games over networks, which will require a bit more effort. Lastly, we exhibit a network fault isolation protocol, which will be a combination of the previous two algorithms, which will be demonstrated in a case study in Section VII. In order to apply the framework of edge-indication vectors, we need an algorithm elevating the asymptotic differentiation achieved in the previous section to an on-line differentiation scheme. Thus, we make the following assumption:

**Assumption 4.** There exists an algorithm $\mathcal{A}$ which receives a model for a diffusively-coupled network $(G, \Sigma, \Pi)$ and a conjectured limit $y^*$ as input, and takes measurements of the network in-run. The algorithm stops and declares “no” if and only if the network does not converge to $y^*$, and otherwise runs indefinitely. The algorithm $\mathcal{A}$ is called a convergence assertion algorithm.

In the language of Section III this is a fault-monitoring system that never gives false positives nor false negatives. For now, we assume such algorithm exists. We will discuss this assumption in Section VI.
A. Detecting Network Faults

We first focus on Problem 1. To tackle the problem, we use the notion of edge-indication vectors from Section [IV]. Suppose we have MEIP agents \( \{ \Sigma_i \} \). We first take any output-strictly MEIP controllers \( \{ \Pi_i \} \) solving the classical synthesis problem, i.e., forcing the closed loop system to converge to \( y^* \) (see Theorem 3). As we noted, if \( w \in \mathbb{R}^{|E|} \) lies in the kernel of \( E_y P_y \), then the solution of the following equations is the same:

\[
k^{-1}(y) + E_y \gamma(E_y^T y) = -E_y P_y w, \quad k^{-1}(y) + E_y \gamma(E_y^T y) = 0.
\]

Thus, if \( w \) lies in \( \ker(E_y P_y) \), running the interaction protocol \( (\Pi, w) \) does not change the steady-state output of the system. However, by Theorem 5, a random vector in \( \ker(E_y P_y) \) gives a \((\Theta, G)\)-edge-indication vector, as long as \( G \) is 2-connected. Thus, using the interaction protocol \( (\Pi, w) \), where \( w \in \ker E_y P_y \) is chosen randomly, will guarantee that all faulty system would converge to a steady-state different than \( y^* \), with probability 1. Thus, applying the algorithm \( \mathcal{A} \) allows an on-line, finite time distinction between the nominal faultless system and its faulty versions. We explicitly write the prescribed algorithm below:

Algorithm 1 Network Fault Detection in MEIP MAS

1: Find edge controllers \( \{ \Pi_i \}_{i \in \mathcal{E}_G} \) solving the synthesis problem with graph \( G \), agents \( \Sigma \), and control goal \( y^* \) (see Theorem 3 and Remark 4).
2: Find a basis \( \{ b_1, ..., b_k \} \) for the linear space \( \ker E_y P_y \).
3: Pick a Gaussian random vector \( \alpha \in \mathbb{R}^k \) and define \( w = \sum b_i \alpha_i \).
4: Define the interaction protocol as \( (\Pi, w) \).
5: Run the system with the chosen interaction protocol.
6: Implement the algorithm \( \mathcal{A} \) for the system \( (G, \Sigma, (\Pi, w)) \) with limit \( y^* \). Declare a fault in the network if \( \mathcal{A} \) declares that the system does not converge to the prescribed value.

Theorem 6 (Network Fault Detection). Suppose that \( n \) agents \( \{ \Sigma_i \} \) and an underlying graph \( G \) are given, and that \( \{ \Sigma_i \} \) satisfy Assumption 2. Suppose furthermore that \( G \) is 2-connected. Then, with probability 1, Algorithm 7 synthesizes an interaction protocol \( (\Pi, w) \) solving Problem 1, i.e., the algorithm satisfies the following properties:

i) If no faults occur in the network, the output of the closed-loop system converges to \( y^* \).
ii) If any number of faults occurs in the network, the algorithm detects their existence.

Proof. Follows from the discussion preceding Algorithm 1. Namely, Theorem 3 assures that \( w \) is a \((\Theta, G)\)-edge-indication vector, so long that \( G \) is 2-connected. In other words, the output of the closed-loop system with graph \( G \) converges to \( y^* \), and for any graph \( G \neq \mathcal{H} \in \Theta \), the output of the closed-loop system with graph \( \mathcal{H} \) converges to a value different from \( y^* \).

It remains to show that the algorithm declares a fault if and only if a fault occurs. If no faults occur, \( \mathcal{A} \) never declares a fault, and the same is true for Algorithm 1. On the contrary, assume any number of faults occur in the network, and let \( \mathcal{H} \) be the current underlying graph. The output of the closed loop system converges to \( y \neq y^* \), so \( \mathcal{A} \) eventually stops and declares that the network does not converge to the conjectured limit. Thus Algorithm 1 declares a fault. \( \square \)

B. Multi-Agent Synthesis in the Presence of an Adversary

Consider the following 2-player game. Both players are given the same \( n \) SISO agents \( \Sigma_1, \cdots, \Sigma_n \), the same graph \( G \) on \( n \) vertices and \( m \) edges, and the same vector \( y^* \in \mathbb{R}^n \). There is also a server that can measure the state of the agents at certain intervals, and broadcast a single message to all agents once. The planner acts first, and designs a control scheme for the network and the server. The adversary acts second, removing at most \( r \) edges from \( G \). The system is then run. The planner wins if the closed-loop system converges to \( y^* \), and the adversary wins otherwise. We show that the planner can always win by using a strategy stemming from edge-indication vectors, assuming the agents are MEIP.

Algorithm 2 Planner Strategy in Adversarial Multi-Agent Synthesis Game with MEIP agents - Synthesis

1: Define \( N = \sum_{\ell=0}^r \binom{n}{\ell} \). Let Graphs be an array with \( N \) entries, and let \( j = 1 \).
2: for \( \ell = 0, \cdots, r \) do
3: \( i = 1 \leq i_1 < i_2 < \cdots < i_\ell \leq m \) do
4: Insert the graph \( H = G - \{ e_{i_1}, \cdots, e_{i_\ell} \} \) to the \( j \)-th entry of Graphs. Advance \( j \) by 1.
5: end for
6: end for
7: Define two arrays Controllers,SSLimits of length \( N \).
8: Choose \( w \) as Gaussian random vector of length \( m \).
9: for \( j = 1, \cdots, N \) do
10: Take any edge controllers \( \{ \Pi_i \}_{i \in \mathcal{E}_G} \) satisfying Assumption 1.
11: Compute the steady-state limit of the network with agents \( \Sigma_i \), underlying graph \( \text{Graphs}(j) \), and interaction protocol \( (\Pi, w) \). Insert the result into SSLimits(j).
12: Solve the synthesis problem for agents \( \Sigma \) and underlying graph \( \text{Graphs}(j) \) as in Theorem 3 and Remark 1. Insert the result into Controllers(j).
13: end for
the synthesis algorithm has polynomial time complexity. Otherwise, the time complexity is \( O(n^c) \) for some universal constant \( c > 0 \). Furthermore, the size of the message broadcasted by the server is \( O(r \log n) \).

Proof. Suppose the adversary changed the underlying graph to \( \mathcal{H} \), which has entry \( j \) in Graphs. We note that Assumption 1 assures that \( \mathcal{A} \) never declares a fault if and only if the closed-loop system converges to the conjectured steady-state, and that \( w \) is a \( \Theta \)-edge-indication vector by Theorem 2. Thus, the \( j \)-th instance of convergence assertion protocol can never return a fault, and all other instances must eventually declare a fault. Thus, the server correctly guesses the underlying graph. It then broadcasts the index \( j \) to the agents, allowing them to change the interaction protocol and run the solution of the synthesis problem with desired output \( y^* \) and underlying graph \( \mathcal{H} \). Thus the network will converge to \( y^* \), and the planner wins.

We now move to time complexity. Note that \( N = O(mn^r) \). The first for-loop has \( N \) iterations, each of takes no more than \( O(mn) \) actions done (where a graph is saved in memory by its incidence matrix, which is of size \( m \times n \)). Thus the first for-loop takes a total of \( O(mn^{r+1}n) \) time. The second for-loop is a bit more complex. It solves the synthesis problem for \( \mathcal{H} \) by solving an equation of the form \( \mathcal{E}_{\mathcal{H}}v = v_0 \) for some known vector \( v_0 \) and an unknown \( v \). This can be done using least-squares, which takes no more than \( O(\max(m,n)^3) \) time. As for finding the communication complexity, this can be done by solving a convex network optimization problem \([25]\) Problem (OPP), which takes a polynomial amount of time in \( n,m \) (e.g. via gradient descent). Recalling that \( m \leq \binom{n}{2} = O(n^2) \), we conclude that if \( r \) is bounded, the total time used is polynomial in \( n \). Moreover, if \( r \) is unbounded, the bottleneck is the first for-loop which takes \( O(mn^{r+1}n) \) time. Plugging \( m \leq n^2 \) gives a bound on the time complexity of the form \( O(n^{2r}) \), as \( n^3 = O(n^{2r}) \). As for the communication complexity, the server broadcasts a number between 1 and \( N \), so a total of \( O(\log_2 N) \) bits are needed to transmit the message. Plugging in \( N = O(mn^r) \) gives that the number of bits needed is \( O(r \log_2 m) = O(r \log n) \).

### Algorithm 3 Planner Strategy - In-Run Protocol for Agents

1. Run the interaction protocol \((\Pi,w)\).
2. When a message \( j \) is received, run the interaction protocol described by \text{Controllers}(j).

### Algorithm 4 Planner Strategy - In-Run Protocol for Server

1. Define \( \text{HasFaulted} \) as an array of zeros of size \( N \).
2. while \( \text{HasFaulted} \) has at least two null entries do
3. Run \( N \) instances of the algorithm \( \mathcal{A} \) simultaneously, with conjectured steady-states limits from SSLimits.
4. for \( j = 1 \) to \( N \) do
5. if The \( j \)-th instance declared “no” then
6. Change the value of \( \text{HasFaulted}(j) \) to 1.
7. end if
8. end for
9. end while
10. Find the index \( j \) such that \( \text{HasFaulted}(j) = 0 \). Broadcast the message \( j \) to the agents.

extra faults have been found, and the current underlying graph is not yet known. The first phase can be solved by using the network fault detection Algorithm 1 as long as the current underlying graph is 2-connected. The second phase can be solved by the pre-broadcast stage of the planner strategy described in Algorithms 2, 3, and 4.

The main issue that remains is what happens if the underlying graph changes again in the exploratory phase, i.e. we entered the exploratory phase with underlying graph \( \mathcal{H}_1 \), but it changed to \( \mathcal{H}_2 \) before we identified that graph. In the exploratory phase, we run an instance of \( \mathcal{A} \) on all of the possible graphs simultaneously, until all but one instance declared a fault. If the instance related to \( \mathcal{H}_2 \) has not declared a fault yet, it will not declare a fault from now on, unless another fault occurs before the end of the exploratory phase. If the same instance has already declared a fault, we have a problem - all other instances will eventually also declare a fault, and there are two options in this case.

The first option is that one instance will declare a fault last, i.e. there is a time in which all but one instance have declared a fault. In this case, we identify the graph as some \( \mathcal{H}_3 \). When we return to the stable phase and run the interaction protocol \((\Pi,w)\) corresponding to \( \mathcal{H}_3 \), a fault will be declared and we will return to the exploratory phase. This is because \( w \) was synthesized as a \((\Theta,\mathcal{H}_3)\)-edge-indication vector, meaning the de-facto steady-state limit (with graph \( \mathcal{H}_2 \)) will be different than the conjectured steady-state limit (with graph \( \mathcal{H}_3 \)), and \( \mathcal{A} \) will declare a fault. The second option is that the last few instances of the convergence assertion protocol declare a fault simultaneously, i.e. there is a time in which all instances have declared a fault, which is dealt with by restarting the exploratory phase. We get the synthesis algorithm and in-run protocol presented in Algorithms 3 and 4. We claim these solve Problem 2. We prove:

**Theorem 8.** Let \( \Sigma_1,\ldots,\Sigma_n \) be agents satisfying Assumption 7 and let \( \mathcal{G} \) be a \( k \)-connected graph for \( k \geq 3 \) on \( n \) vertices and \( m \) edges. Then, with probability 1, Algorithms 3 and 4 run with \( r = k - 2 \), solve Problem 2 for up to \( r \) faults.

**Proof.** We refer to steps 2 to 3 of Algorithm 4 as the stable phase of the algorithm, and to steps 4 to 13 as the exploratory phase. As the number of faults is no bigger than \( r = k - 2 \), the underlying graph remains 2-connected throughout the run. We
We first explain why the claims hold with probability 1. Claims 1 and 2 follow from Theorem 6 with probability 1, as the underlying graph \( G \) is always 2-connected. Claim 4 holds with probability 1, as follows from Theorem 7. As for Claim 3, if no faults occur throughout the instance of the exploratory phase, then it eventually ends by Claim 4. If faults do occur throughout the run, the discussion above shows that at some time, all (except possibly one) instances of the convergence assertion protocol have declared a fault. If all of them declared a fault, we start another instance of the exploratory phase, and otherwise, we move to the stable phase. In either case, the instance of the exploratory phase ends, and Claim 3 is proved.

We now explain how the theorem follows from these claims. Suppose a total of \( \ell \leq r \) faults occur throughout the run. Let \( T < \infty \) be the time at which the last fault occurs. We look at the phase of the algorithm at times \( t > T \), and show that in either case the system must converge to \( y^* \).

i) If we arrive at the stable phase with and the conjectured graph \( H \) is the true underlying graph, then the system converges to \( y^* \) (Claim 1).

ii) If we start an instance of the exploratory phase, it eventually ends (Claim 3) and the stable phase starts, in which the conjectured graph \( H \) is the true underlying graph (Claim 4). By i), the system converges to \( y^* \).

iii) If we are in the stable phase, but the conjectured graph \( H \) is different from the true underlying graph, we eventually start an exploratory phase (Claim 3). Thus, the system converges to \( y^* \) by ii).

iv) Lastly, we could be in the middle of an instance of the exploratory phase. In that case, the instance eventually ends (Claim 3), after which we either apply a new instance of the exploratory phase, or the stable phase. In both cases, we can use either i), ii) or iii) and conclude the system must converge to \( y^* \).

\[ \square \]

Remark 2. We can use a similar protocol to isolate more complex faults. We consider the collection of subgraphs \( \mathcal{H} \) of \( G \) in which there is a set of vertices of size \( \leq r \), so that each edge in \( G - \mathcal{H} \) touches at least one vertex in the set. This observation allows us to offer similar network fault detection and isolation algorithms for more complex types of faults. For example, we can consider a case in which each agent communicates with all other agents by a single transceiver, and if it faults, then all edges touching the corresponding vertex are removed from the graph. We can even use a hybrid fault model, in which faults correspond to certain subsets of edges touching a common vertex are removed from the graph. For example, suppose there are two distant groups of agents. Agents in the same group are close, and communicate using Bluetooth communication. Agents in different groups are farther, and communicate using Wi-Fi (or broadband cellular communication). When an agent’s Bluetooth transceiver faults, all inter-group edges are removed, and when the Wi-Fi transceiver faults, all intra-group edges are removed.

VI. ONLINE ASSERTION OF NETWORK CONVERGENCE

In the previous section, we used the notion of edge-indication vectors, together with Assumption 4 to suggest
algorithms for network fault detection and isolation. The goal in this section is to propose algorithms of satisfying Assumption 4. This will be achieved by using convergence estimates, relying on passivity. We revisit a result from [25].

**Proposition 3** (25). Let \((u, y, \zeta, \mu)\) be a steady-state of \((G, \Sigma, \Pi)\) of the form (4). Suppose that the agents \(\Sigma_i\), with state \(x_i\), are passive with respect to \((u_i, y_i)\) with passivity index \(\rho_i \geq 0\), and that the controllers \(\Pi_e\), with state \(\eta\), are passive with respect to \((\zeta_e, \mu_e)\), with passivity index \(\nu_e \geq 0\). Let \(S(x_i)\) and \(W_e(\eta)\) be the agents' and the controllers' state functions, respectively. Then \(S(x, \eta) = \sum_{x \in X} S(x) + \sum_{\eta \in E} W_e(\eta)\) is a positive-definite \(C^1\)-function, which nulls only at the steady-states \((x, \eta)\) corresponding to \((u_i, y_i)\) and \((\zeta_e, \mu_e)\), and satisfying the inequality:

\[
\frac{dS}{dt} \leq - \sum_{i \in V} \rho_i (y_i(t) - y_i)^2 - \sum_{e \in E} \nu_e (\mu_e(t) - \mu_e)^2. \tag{6}
\]

**Proof.** The proof follows immediately from \(S_i, W_e\) being positive-definite \(C^1\)-functions nulling only at \(x_i, \eta_e\), by summing the following inequalities:

\[
\frac{dS_i}{dt} \leq (u_i(t) - u_i)(y_i(t) - y_i) - \rho_i (y_i(t) - y_i)^2,
\]

\[
\frac{dW_e}{dt} \leq (\mu_e(t) - \mu_e)(\zeta_e(t) - \xi_e) - \nu_e (\mu_e(t) - \mu_e)^2,
\]

and using the equality \((u(t) - u)^T(y(t) - y) = -(\mu(t) - \mu)^T\xi_e^T(y(t) - y) = -(\mu(t) - \mu)^T(\zeta(t) - \zeta).

The inequality (6) can be thought of as a way to check that the system is functioning properly. Indeed, we can monitor \(x, y, \eta, \mu\), and check that the inequality holds. If it does not, there must have been a fault in the system. This idea has a few drawbacks, linked to one another. First, as we commented in Subsection III-A in some networks, the controller state \(\eta_e(t)\) can be defined only for existing edges, so using \(\eta(t)\) requires us to know the functioning edges, which is absurd. Thus, in some cases, we must use Assumption 2. Second, in practice, even if we have access to \(x\), we cannot measure it continuously. Instead, we measure it at certain time intervals. One can adapt (6) to an equivalent integral form:

\[
S(x(t_{k+1}), \eta(t_{k+1})) - S(x(t_k), \eta(t_k)) \leq \int_{t_k}^{t_{k+1}} \left( \sum_{i \in V} \rho_i \Delta y_i(t)^2 + \sum_{e \in E} \nu_e \Delta \mu_e(t)^2 \right) dt, \tag{7}
\]

where \(\Delta y_i = y_i(t) - y_i\) and \(\Delta \mu_e = \mu_e(t) - \mu_e\). However, this gives rise to the third problem - unlike the function \(S\), we cannot assure that the functions \((y_i(t) - y_i)^2\) and \((\mu_e(t) - \mu_e)^2\) (or their sum) is monotone. Thus, we cannot compute the integral appearing on the right-hand side of the inequality.

We present two approaches to address this problem. First, we try and estimate the integral using high-rate sampling, by linearizing the right hand side of (7) and bounding the error. Second, we try to bound the right-hand side as a function of \(S\), resulting in an inequality of the form \(\dot{S} \leq -F(S)\), which will give a convergence estimate.

A. Assenting Convergence Using High-Rate Sampling

Consider the inequality (7), and suppose \(t_{k+1} - t_k = \Delta t_k\) is very small, so the functions \(y_i(t) - y_i\) and \(\mu_e(t) - \mu_e\) are roughly constant in the time period used for the integral. More precisely, recalling that \(y = h(x)\) and \(\mu = \phi(\eta, E_0^y, y)\), and assuming these functions are differentiable near \((x(t_k), \eta(t_k))\), we expand the right-hand side of (7) to a Taylor series,

\[
\int_{t_k}^{t_{k+1}} \left( \sum_{i \in V} \rho_i \Delta y_i(t)^2 + \sum_{e \in E} \nu_e \Delta \mu_e(t)^2 \right) dt = (8)
\]

\[
\sum_{i \in V} \rho_i \Delta y_i(t_k)^2 + \sum_{e \in E} \nu_e \Delta \mu_e(t_k)^2 \Delta t_k + O(\Delta t_k^2).
\]

We wish to give a more explicit bound on the \(O(\Delta t_k^2)\) term. We consider the following function \(G\), defined on the interval \([t_k, t_{k+1}]\) by the formula

\[
G(t) = \sum_{i \in V} \rho_i (y_i(t) - y_i)^2 + \sum_{e \in E} \nu_e (\mu_e(t) - \mu_e)^2. \tag{9}
\]

The equation (8) is achieved from the approximation \(G(t) = G(t_k) + O(t - t_k)\) which is true for differentiable functions. Using LaGrange’s mean value theorem for \(t \in [t_k, t_{k+1}]\), we find some point \(s \in (t, t_{k+1})\) such that \(G(t) = G(t_k) + \frac{dG}{dt}(s)(t - t_k)\). If we manage to bound the time derivative \(\frac{dG}{dt}\) in the interval \([t_k, t_{k+1}]\), we would find a computational way to assert convergence. By the chain rule, the time derivative of \(G\) is given by

\[
\frac{dG}{dt} = \sum_{i \in V} \rho_i (y_i(t) - y_i) \dot{y}_i + \sum_{e \in E} \nu_e (\mu_e(t) - \mu_e) \dot{\mu}_e. \tag{10}
\]

In order to compute the time derivative of \(y_i, \mu_i\), we recall that both are functions of \(x\) and \(\eta\), namely \(y = h(x)\) and \(\mu = \phi(\eta, E_0^y, y) = \phi(\eta, E_0^y, h(x))\). Thus, we have that

\[
\begin{align*}
\dot{y} &= \nabla_x h(x(t)) \dot{x} \\
\dot{\mu} &= \nabla_{\eta} \phi(\eta(t), \zeta(t)) \dot{\eta} + \nabla_{\eta} \phi(\eta(t), \zeta(t)) E_0^y \nabla h(x(t)) \dot{x},
\end{align*}
\]

where \(\zeta(t) = E_0^y h(x(t))\), \(\dot{x} = f(x, u) = f(x, -E_0^y \phi(\eta, \zeta))\), and \(\dot{\eta} = \psi(\eta, \zeta) = \psi(\eta, E_0^y h(x))\). Thus, we can write the time derivative of \(G\) as a continuous function of \(x(t), \eta(t)\), as we plug the expressions for \(\dot{y}, \dot{\mu}\) into (10). However, we do not know the value of \(x(t), \eta(t)\) between measurements. To tackle this problem, notice that we have some information on where \(x(t), \eta(t)\) can lie. Namely, equation (3) shows that \(S(x(t), \eta(t))\) is a monotone descending function. Thus, we know that \(x(t), \eta(t)\) lie in the set \(B = \{x, \eta : S(x, \eta) \leq S(x(t_k), \eta(t_k))\}\). More precisely, we show the following.

**Proposition 4.** Assume the functions \(h_i, f_i, \phi_i, \psi_i\) are all continuously differentiable. Then, for any time \(t \in [t_k, t_{k+1}]\), the following inequality holds:

\[
\left| \frac{dG}{dt}(t) \right| \leq (\rho_\delta M_{\delta, 0} M_y + \nu_\delta M_{\delta, 0} M_\mu) M_k + \nu_\delta M_{\delta, \mu} M_{\mu, \delta} M_y,
\]

where \(M_\delta = \max_{(x, \eta) \in B} \| f(x, -E_0^y \phi(\eta, E_0^y h(x))) \|, M_\delta = \max_{(x, \eta) \in B} \| \psi(\eta, E_0^y h(x)) \|, M_k = \max_{(x, \eta) \in B} \| \nabla h(x) \|, M_{\mu, \delta} = \max_{(x, \eta) \in B} \| \nabla \phi(\eta, E_0^y h(x)) \|, M_{\mu, \eta} = \max_{(x, \eta) \in B} \| \nabla \phi(\eta, E_0^y h(x)) \|, M_{\mu, \eta} = \max_{(x, \eta) \in B} \| \nabla \phi(\eta, E_0^y h(x)) \|, M_{\mu, \eta} = \max_{(x, \eta) \in B} \| \nabla \phi(\eta, E_0^y h(x)) \|, \mu_k = \max_{(x, \eta) \in B} \| \eta - h(x) \|. \]
Proof. We fix some \( t \in [t_k, t_{k+1}] \), so that \( (x(t), y(t)) \in \mathbb{B} \). We use the expressions for \( \dot{x}, \dot{y}, \dot{\mu} \) found in (11). First, the conditions \( |\dot{x}| \leq M_x \) and \( |\dot{\mu}| \leq M_\mu \) are obvious. Equation (11) shows that \( |\dot{y}| \leq M_y \dot{y}_x + M_\nu \dot{\mu}_y \). By using Cauchy-Schwartz inequality on (10), we obtain \( |\dot{\mu}| \leq M_\mu \dot{\mu}_x + M_\mu \dot{\mu}_y \). Thus (7) implies that

\[
S(x(t_{k+1})) - S(x(t_k)) \leq \left( \sum_{i \in \mathbb{E}} p_i \Delta y_i(t_k)^2 + \sum_{e \in \mathbb{E}} v_e \Delta \mu_e(t_k)^2 \right) \Delta t_k + \frac{M}{2} \Delta t_k^2,
\]

where

\[
M = (p_x M_\Delta y y + \nu_x M_\Delta y \mu_x M_x + \nu_x M_\Delta \mu \mu_y M_y).
\]

Proof. Recall that \( G(t) = \sum_{i \in \mathbb{E}} p_i (y_i(t) - y_i)^2 + \sum_{e \in \mathbb{E}} v_e (\mu_e(t) - \mu_e)^2 \). By Proposition 4, for every \( t \in [t_k, t_{k+1}] \) we have \( G(t) \leq G(t_{k+1}) + M |t - t_{k+1}| \). Thus (7) implies that \( S(x(t_{k+1})) - S(x(t_k)) \leq G(t_{k+1}) \Delta t_k + \frac{M}{2} \Delta t_k^2 \).

Corollary 1. Fix any two times \( t_k < t_{k+1} \), and consider the notation of Proposition 4. Then the following inequality holds:

\[
S(x(t_{k+1})) - S(x(t_k)) \leq \left( \sum_{i \in \mathbb{E}} p_i \Delta y_i(t_k)^2 + \sum_{e \in \mathbb{E}} v_e \Delta \mu_e(t_k)^2 \right) \Delta t_k + \frac{M}{2} \Delta t_k^2,
\]

where

\[
M = (p_x M_\Delta y y + \nu_x M_\Delta y \mu_x M_x + \nu_x M_\Delta \mu \mu_y M_y).
\]

Proof. It is enough to show that for each \( \epsilon > 0 \) and any \( N_2 > 0 \), there is some \( N > N_1 \) such that \( \sum_{i \in \mathbb{E}} p_i \Delta y_i(t_N)^2 + \sum_{e \in \mathbb{E}} v_e \Delta \mu_e(t_N)^2 < \frac{M}{2} \Delta t_N + \epsilon \). Indeed, suppose this is not the case. Then for any \( k > N_1 \), the right-hand side of (12) is upper-bounded by \( -\epsilon \Delta t_k \). We sum the telescopic series and conclude that for any \( k > N_1 \),

\[
S(x(t_k)) - S(x(t_{N_1})) \leq - \sum_{j=N_1+1}^{k} \epsilon \Delta t_j = -\epsilon (t_k - t_{N_1}),
\]

so \( t_{N_2} \geq t_{N_1} + \epsilon^{-1} S(x(t_{N_1}), y(t_{N_1})) \) implies that \( S(x(t_k), y(t_k)) < 0 \). This is absurd, as \( S \geq 0 \). Thus there must exist some \( N > N_1 \) such that \( \sum_{i \in \mathbb{E}} p_i \Delta y_i(t_N)^2 + \sum_{e \in \mathbb{E}} v_e \Delta \mu_e(t_N)^2 < \frac{M}{2} \Delta t_N + \epsilon \). The second part follows from (13) and the demand that \( S(x(t_k)) \geq 0 \).

Proposition 5 can be used for convergence assertion. We can consider the following scheme - begin at time \( t_0 \) and state \( x_0, \eta_0 \). We want to show that \( S(x(t), \eta(t)) \to 0 \). We instead show that \( G(t) \), defined in (9), gets arbitrarily close to 0. As we said, this is enough as \( G(t) \) is a \( C^1 \) non-negative function of the state \( x(t), \eta(t) \) that is only small when \( x(t), \eta(t) \) is close to the steady-state \( (x, \eta) \). We prove:

Theorem 9. Consider the algorithm \( \mathcal{A} \), defined in the following form. Sample the system at times \( t_1, t_2, \ldots \), and check whether the inequality (12) holds. If it does, continue, and if does not, then stop and declare “no”. Then there exists a sequence \( t_1, t_2, \ldots \), depending on the system and the initial conditions, such that \( \mathcal{A} \) satisfies Assumption 4.

Proof. We present the following method of choosing \( t_1, t_2, \ldots \). We first choose \( t_0 = 0 \), an arbitrary \( \delta_1 > 0 \), compute \( M \) as in Proposition 4 and choose \( \Delta t = \frac{\delta_1}{M} \) and \( \varepsilon = \frac{\delta_1}{2M} \). Sample the system at rate \( \Delta t \) until time \( t_{N_1} = t_0 + \varepsilon^{-1} S(x_0, \eta_0) \). The process is then reiterates with \( \delta_{k+1} = \delta_k^{1/2} \) for \( k = 1, 2, \ldots \), giving rates \( \Delta t_k \) and times \( t_{N_k} \). We claim that \( \mathcal{A} \), with this choice of sample times, satisfies Assumption 4. If the diffusively-coupled network \( (G, \Sigma, \Pi) \) converges to \( (x, \eta) \), then Corollary 1 implies the algorithm never stops, as required. It remains to show that if the algorithm never stops, the network converges to the conjectured limit. By the discussion above, and the fact that \( S(x(t), \eta(t)) \) is a monotone descending function, it is enough to show that \( \lim_{k \to \infty} G(t_k) = 0 \). We first show at some point, \( G(t) < \delta_1 \). By choice of \( \Delta t \), if (12) holds at each time, then when we reach time \( t_{N_k} \), we know that at some point, we had \( G(t) \leq \frac{M}{2} \Delta t + \varepsilon = \delta_1 \). Reiterating shows that at some times \( t_k, G(t_k) \leq \delta_k \), where \( \delta_{k+1} = \delta_k^{1/2} \), so \( \lim_{k \to \infty} G(t_k) = 0 \).

The term “High-Rate Sampling” comes from the fact that if \( M \) is not updated when we re-iterate with smaller \( \delta \), then eventually, \( t_{k+1} - t_k \to 0 \), which is impractical in real-world cases. However, we note that the number \( M \) decreases as \( S(x(t), \eta(t)) \) decreases, as shown in Proposition 4. Thus, if \( M \) is updated between iterations, we might have \( \Delta t \to 0 \).

Remark 3. There is a trade-off between the time-steps \( \Delta t \) and the time it takes to find a point in which \( G(t) < \frac{M}{2} \Delta t + \varepsilon \), which is \( t = \frac{S(x(0), \eta(0))}{\varepsilon} \). On one hand, we want larger...
time-steps (to avoid high-rate sampling) and shorter overall times, however, increasing both \( \Delta t \) and \( \varepsilon \) creates a worse eventual bound on \( G(t) \). We can choose both by maximizing an appropriate cost function \( C(\Delta t, \varepsilon) \), monotone in both \( \Delta t \) and \( \varepsilon \), subject to \( \frac{\Delta t}{2} \Delta t + \varepsilon = \delta t, \varepsilon \geq 0, \Delta t \geq 0 \). Choosing \( C(\Delta t, \varepsilon) \) as linear is inadvisable, as the maximizing a linear function with linear constraints always leads to the optimizer being on the boundary, which means either \( \Delta t = 0 \) or \( \varepsilon = 0 \). The choice \( \Delta t = \frac{\delta t}{2} \) and \( \varepsilon = \frac{\delta t}{2} \) mentioned above corresponds to the geometric average cost function \( C(\Delta t, \varepsilon) = \sqrt{\Delta t \varepsilon} \). Other choices of \( C \) can express practical constraints, e.g. relative apathy to large convergence times relative to high-rate sampling should result in a cost function penalizing small values of \( \Delta t \) more harshly than small values of \( \varepsilon \).

B. Asserting Convergence Using Convergence Profiles

For this subsection, we now assume that Assumption 3 holds and that the agents are output-strictly MEIP, i.e., that \( \rho_i > 0 \). Consider \( \mathcal{F} \) and suppose there is a non-negative monotone function \( \mathcal{F} \) such that for any \( t \), the right-hand side of \( \mathcal{F} \) is bounded from above by \( -\mathcal{F}(S) \). In that case, we get an estimate of the form \( \dot{S} \leq -\mathcal{F}(S) \). This is a weaker estimate than \#4, but it has a more appealing discrete-time form,

\[
S(x(t_{k+1})) - S(x(t_k)) \leq - \int_{t_k}^{t_{k+1}} \mathcal{F}(S(x(t))) dt 
\]

(14)

\[
\leq -\mathcal{F}(S(x(t_{k+1}))) \cdot (t_{k+1} - t_k),
\]

where we use the monotonicity of \( \mathcal{F} \) and the fact that \( S(x(t)) \) is monotone non-ascending. Due to Assumption 3 we focus on the elements of the right-hand side of \( \mathcal{F} \) corresponding to the agents, and neglect the ones corresponding to controllers, as \( S \) is now the sum of the functions \( S_i(x_i) \). Because controllers are passive, we have \( \nu_e \geq 0 \), so removing the said term does not change the inequality's validity.

In order to find \( \mathcal{F} \), it is natural to look for functions \( \Omega \) satisfying \( \Omega(S_i) \leq (y_i(t) - y_i) \) for \( \nu_e \geq 0 \), so removing the said term does not change the inequality's validity.

Definition 5. Let \( \Omega : [0, \infty) \rightarrow [0, \infty) \) be any function on the non-negative real numbers. We say that an autonomous system has the convergence profile \( (\rho, \Omega) \) with respect to the steady-state \( (u, y) \) if there exists a \( C^1 \) storage function \( S(x) \) such that the following inequalities hold:

i) \( \frac{dS(x(t))}{dt} \leq (u(t) - u)(y(t) - y) - \rho(y(t) - y)^2 \),

ii) \( \Omega(S(x(t))) \leq (y(t) - y)^2 \).

Example 1. Consider the SISO system \( \dot{x} = -x + u, y = x \), and consider the steady-state input-output pair \( (0, 0) \). The storage function \( S(x(t)) = \frac{1}{2} x(t)^2 \) satisfies

\[
\dot{S}(x(t)) = x(t) \dot{x}(t) = (u(t) - 0)(y(t) - 0) - (y(t) - 0)^2.
\]

Thus \( \Sigma \) has convergence profile \( (1, \Omega) \) for \( \Omega(\theta) = 2\theta \).

More generally, when considering an LTI system with no input-feederthrough, both functions \( S(x) \) and \( (y(t) - y)^2 \) are quadratic in \( x \). Thus there is a monotone linear function \( \Omega \) such that the inequality \( \Omega(S(x(t))) \leq (y(t) - y)^2 \) holds. In particular, a function \( \Omega \) exists in this case. We show that functions \( \Omega \) exist for rather general systems.

Theorem 10. Let \( \Sigma \) be the SISO system of the form \( \dot{x} = -f(x) + q(x)u, y = h(x) \). Suppose \( q \) is a positive continuous function, that \( f/q \) is \( C^1 \) and monotone ascending and that \( h \) is \( C^1 \) and strictly monotone ascending. Let \( (u = f(x)/q(x), y = h(x)) \) be any steady-state input-output pair of the system. Then

i) using the storage function \( S(x) = \int_x^h \frac{h(x') - h(x)}{q(x')} dx \), the system \( \Sigma \) has the convergence profile \( (\rho, \Omega) \) for a strictly ascending function \( \rho \) and \( \rho = \inf_x \frac{f(x) - h(x)}{q(x)} \geq 0 \);

ii) suppose there exists some \( \alpha > 0 \) such that the limit \( \lim_{x \rightarrow \infty} \frac{|h(x) - h(x')|}{|x|^{\alpha+1}} \) exists and is finite. Then the limit \( \lim_{x \rightarrow 0} \frac{\Omega(x)}{x^{\alpha}} \) exists and is finite, where \( \beta = \frac{2\alpha}{\alpha+1} \). In other words, if \( h \) behaves like a power law near \( 0 \), then \( \Omega \) behaves like a power law near \( 0 \).

Proof. The passivation inequality follows from Theorem 1 so we focus on constructing the function \( \Omega \). For every \( \theta \geq 0 \), we define the set \( \Lambda_\theta = \{x \in \mathbb{R} : (h(x) - h(x'))^2 \leq \theta \} \). We want that \( x \in \Lambda_\theta \) would imply that \( \Omega(S(x)) \leq \theta \). Because \( h \) is continuous and monotone, it is clear that \( \Lambda_\theta \) is an interval containing \( x \). Now, let \( \omega \) be the function on \( [0, \infty) \) defined as \( \omega(\theta) = \sup_{x \in \Lambda_\theta} S(x) \). We note that \( \omega \) can take infinite values (e.g. when \( h \) is bounded, but \( S \) is not). However, we show that the restriction of \( \omega \) on \( \{\theta : \omega(\theta) < \infty\} \) is strictly monotone. If we show that this claim is true, then \( \omega \) has an inverse function which is also strictly monotone. Define \( \Omega = \omega^{-1} \) as the strictly monotone inverse function. By definition, for any \( x \in \mathbb{R} \) we have that \( x \in \Lambda_\theta \) for \( \theta = (h(x) - h(x'))^2 \), so \( S(x) \leq \omega(\theta) \). Thus \( \Omega(S(x)) \leq \Omega(\omega(\theta)) = \theta = (h(x) - h(x'))^2 \), concluding the first part of the proof.

We now prove that the restriction of \( \omega \) on \( \{\theta : \omega(\theta) < \infty\} \) is strictly monotone. It is clear that if \( 0 \leq \theta_1 < \theta_2 \) then the interval \( \{x : (h(x) - h(x'))^2 \leq \theta_1\} = \Lambda_{\theta_1} \) is strictly contained in the interval \( \{x : (h(x) - h(x'))^2 \leq \theta_2\} = \Lambda_{\theta_2} \), as \( h \) is strictly monotone. Moreover, it is clear that \( S \) is strictly ascending in \( [x, \infty) \) and strictly descending in \( (-\infty, x] \), as the function \( \frac{h(x) - h(x')}{q(x')} \) is positive on \( (x, \infty) \) and negative on \( (-\infty, x) \). Thus we have \( \omega(\theta_1) < \omega(\theta_2) \), unless \( \omega(\theta_1) = \infty \), which is what we wanted to prove.

We now move to the second part of theorem, in which we show that if \( h \) behaves like a power law near \( x \), then \( \Omega \) behaves like a power law near zero. We use big-O notation (in the limit \( x \rightarrow x \)). By assumption and strict monotonicity of \( h \), we have

\[
h(x) - h(x) = C\text{sgn}(x-x)|x-x|^\alpha + o(|x-x|^\alpha)
\]

for some constant \( C > 0 \), implying \( (h(x) - h(x'))^2 = C^2|x-x'|^{2\alpha} + o(|x-x'|^{2\alpha}) \). By definition, we conclude that for \( \theta > 0 \) small enough, \( \Lambda_\theta \) is an interval centered at \( x \) and has radius \( \theta^{1/2\alpha} \). Thus \( \Omega(\theta) = \theta \). We recall that \( S(x) = \int_x^h \frac{h(x') - h(x)}{q(x')} dx \). We write \( q(x) = q(x) + o(1) \) as \( q \) is continuous, so \( \Omega(\theta) \) implies that

\[
\frac{h(x) - h(x)}{q(x)} = \frac{1}{q(x)} (C\text{sgn}(x-x)|x-x|^\alpha + o(|x-x|^\alpha))
\]

Thus, \( S(x) = \frac{C}{q(x)(\alpha+1)}|x-x|^\alpha + o(|x-x|^\alpha+1) \). We now
compute \( \omega(\theta) \) by definition, using our characterization of \( \delta_0 \):

\[
\omega(\theta) = \max_{x \in \mathbb{R}} S(x) = \max_{x \in \mathbb{R}} \left( \frac{C |x - x|^{\alpha+1}}{q(x)(\alpha + 1)} + o(|x - x|^{\alpha+1}) \right)
\]

\[
= \left( \frac{\theta^{2}}{C^{1/\alpha}} \right)^{\alpha+1} + o \left( \left( \frac{\theta}{C} \right)^{\alpha+1} \right) = (D + o(1)) \theta^{\frac{\alpha+1}{\alpha}}
\]

for \( D = \frac{1}{q(x)(\alpha + 1)} \sum_{i} \omega_i > 0 \). Thus, the inverse function \( \Omega(\theta) \) is given as \( \Omega(\theta) = (D - \frac{\alpha}{\alpha+1} - o(1)) \theta^{\frac{\alpha}{\alpha+1}} \), as desired. \( \square \)

**Example 2.** Consider a function with \( q(x) = 1 \), \( h(x) = \sqrt{x} \) and a steady state \( u = x = y = 0 \). \( h(x) \) behaves like a power law with power \( \alpha = \frac{1}{2} \). Part ii) of Theorem 10 implies that \( \Omega \) also behaves like a power law, with power \( \beta = \frac{2}{\alpha+1} = \frac{1}{2} \).

We exemplify the computation of \( \Omega \) as done in the proof, and show it behaves like a power law with \( \beta = \frac{1}{2} \), as forecasted by the theorem. Indeed, \( S(x) = \int_{0}^{x} \sqrt{\theta} \, dt = \frac{2}{3} x^{3/2} \), and \( (h(x) - h(0))^2 = \Omega(S(x)) \).

**Remark 4.** Theorem 10 gives a prescription to design the function \( \Omega \). However, some steps, namely the inversion of \( \omega \), are computationally hard. For example, if \( h(x) = 1 - e^{-x} \) and \( q(x) = 1 \), then \( \omega(\theta) = \log \left( 1 - e^{-\theta} \right) \).

**Proposition 6.** Let \( \rho_1, \ldots, \rho_n \) be any positive numbers, and let \( \Omega_1, \ldots, \Omega_n : [0, \infty) \to [0, \infty) \) be any \( C^1 \) strictly monotone functions such that \( \Omega_i(\theta_i) = 0 \) only at \( \theta_i = 0 \). Suppose further that for any \( i \) there exist some \( \beta_i > 0 \) such that the limit \( \lim_{\delta \to 0} \frac{\Omega_i(\theta_i)}{\delta_i} \) exists and is positive. Define \( \Omega_\star : [0, \infty) \to [0, \infty) \) as \( \Omega_\star(\theta) = \min_i \Omega_i(\theta) \).

The proof of the proposition can be found in the appendix.

**Corollary 2.** Let \( S_1, \ldots, S_n \) be the storage functions of the agents, let \( S = \sum S_i \), and let \( \Omega_1, \ldots, \Omega_n \) be \( C^1 \) strictly monotone functions such that \( \Omega_i(\theta_i) = 0 \) only at \( \theta_i = 0 \). Suppose that for any \( i \) there exist some \( \beta_i > 0 \) such that the limit \( \lim_{\delta \to 0} \frac{\Omega_i(\theta_i)}{\delta_i} \) exists and is positive. Moreover, Suppose that \( \hat{S} \leq \sum \rho_i \Omega_i(S_i) \). Then for every bounded set \( B \subset \mathbb{R}^n \) there exists a constant \( C > 0 \) such that for any trajectory of the closed-loop system with initial condition in \( B \), the inequality \( \hat{S}(t) \leq -C \cdot \Omega_\star(S) \) holds, where \( \Omega_\star(\theta) = \min_i \Omega_i(\theta) \).

**Proof.** Use \( \theta_i = S_i \) and \( D = S(x(0)) \) in Proposition 6. \( \square \)

Proposition 6 and Corollary 2 show that an inequality of the form (14) can be achieved, so long the functions \( \Omega_i \) from Theorem 10 “behave nicely” around \( 0 \), namely do not grow faster nor slower than a power law. This condition is very general, and only excludes pathologies as \( \Omega(\theta) = \frac{1}{\log(1/\theta)} \), growing faster than any power law, and \( \Omega(\theta) = \exp(-1/\theta^2) \), growing slower than any power law. Theorem 10 shows that if \( h \) behaves like a power law near \( x \), then so does \( \Omega \), so pathological functions \( \Omega_\star \) can only come from pathological measurement functions \( h_i \). We show it is enough to check the discretized equation (11) to assert convergence.

**Proposition 7.** Let \( \Omega_\star : [0, \infty) \to [0, \infty) \) be any continuous function such that \( \Omega_\star(\theta) = 0 \) only at \( \theta = 0 \). Let \( S(x) \) be any time-dependent monotone decreasing function such that \( S : [0, \infty) \to [0, \infty) \). Let \( t_1, t_2, t_3, \ldots \) be any unbounded sequence of times such that \( \max_{i \leq n} (t_{k+1} - t_k) > 0 \), and suppose that for every \( k \), the inequality \( \hat{S}(t_{k+1}) - S(t_k) \leq -\Omega_\star(S(t_{k+1}))) \) holds. Then \( \hat{S}(t) \to 0 \) as \( t \to \infty \).

The proof of the proposition can be found in the appendix.

We want to use \( \hat{S}(t) = S(x(t)) \). The results above suggest an algorithm for convergence assertion.

**Algorithm 7 Convergence Assertion using Convergence Profile**

**Input:** A diffusively-coupled network \((G, \Sigma, \Pi)\), an initial condition \( x(0) \) and a conjectured steady-state \( \hat{x} \).

1. Let \( S_i(x_i) = \int_{x_i}^{x_i+1} \frac{h(x_i) - h(x_i)}{q(x_i)} \, dx_i \), \( S(x) = \sum S_i(x_i) \).
2. Use Theorem 10 Proposition 6 and Corollary 2 to find a function \( \Omega \) such that \( \hat{S} \leq -\Omega(S) \) for all times \( t \), with initial condition \( x(0) \).
3. Choose \( \delta_0 = S(x(0)) \) and \( t_0 = 0 \).
4. for \( k = 0, 1, 2, 3, \ldots \) do
5. Define \( \delta_{k+1} = \delta_k/2 \).
6. Define \( M = \min_{x \in S(x) \geq \delta_{k+1}} \Omega(S(x)) \).
7. Sample the system at some \( t_{k+1} > t_k + \frac{S(x_k)}{M} \).
8. if \( S(x(t_{k+1})) - S(x(t_k)) \leq -\Omega(S(x(t_{k+1}))) \) then
9. Stop and return “no”;
10: end if
11: end for

**Theorem 11.** Algorithm 7 taking the system \((G, \Sigma, \Pi)\), the initial state \( x(0) \), and the conjectured steady-state \( \hat{x} = h^{-1}(y) \) as input, satisfies Assumption 2.

**Proof.** We denote the true limit of the system \((G, \Sigma, \Pi)\) by \( x \). We first assume the algorithm never stops, and show that \( \hat{x} = x \). We show that \( S(x(t)) \leq \delta_k \), which would suffice as \( \delta_k \to 0 \) and \( S(t) \to 0 \) implies that \( x(t) \to \hat{x} \), and thus \( x = \hat{x} \). Suppose, heading toward contradiction, that \( S(x(t)) \leq \delta_k \). Then \( \Omega(S(x(t))) \geq M \), meaning that the right-hand side
of the checked inequality is larger than $-S(x(t_k))$. Thus, if the inequality holds then $S(x(t_{k+1})) < 0$, which is absurd. Thus $S(x(t_k)) \leq \delta_k$, and $\hat{x} = x$. On the contrary, if the conjectured limit $\hat{x}$ is the true limit of the network, then Theorem 10 Proposition 1 and Corollary 2 show that $S(x(t_{k+1})) - S(x(t_k)) \leq -\Omega(S(x(t_{k+1}))(x(t_{k+1}) - x(t_k)))$ always holds, so the algorithm never stops, as expected. □

Remark 5. Proposition 7 shows we can take any sample times $t_k$ such that $\liminf_{k \to \infty} (t_k - t_{k-1}) > 0$, and still get a valid convergence assertion algorithm. The suggested algorithm gives extra information, as it also bounds the distance of $x(t_k)$ from $x$. Another way to choose $t_k$ is to use the solution of the ODE $\dot{\hat{S}} = -\Omega(\hat{S})$ with $\hat{S}(t_0) = S(x_0)$. Let $t_k$ be the earliest time in which $S(t_k) \leq \delta_k$. The inequality $\dot{\hat{S}} \leq -\Omega(S)$ assures that $S(x(t_k)) \leq \delta_k$. This method is more demanding, as the minima $M$ computed before can be stored in a table, but the solution to the ODE must be computed on-line.

Remark 6. Although we can prove convergence with this method using very seldom measurements, we should still sample the system at a reasonable rate. This is because we want to detect faults as soon as possible. If we sample the system in too large intervals, we will not be able to sense a fault until a large amount of time has passed.

We conclude this section with a short discussion about the perks and drawbacks of the two presented convergence assertion methods. The convergence profile method allows the designer to sample the system at any desired rate, allowing one to prove convergence using very seldom measurements. Moreover, it gives certain rate of convergence guarantees before running the system. On the contrary, the high-rate sampling method can require a long time to assert convergence to a $\delta$-ball around the desired steady-state, unless one is willing to increase the sampling rate, perhaps arbitrarily. However, its main upshot over the convergence profile method is that we need not assume that Assumption III holds, and that the method is computationally easier, as one can avoid function inversion needed to compute $\Omega$.

VII. CASE STUDY: VELOCITY COORDINATION IN VEHICLES WITH DRAG

Consider a collection of $n = 20$ vehicles trying to coordinate their velocity. Each vehicle is modeled as a double integrator $G(s) = 1/s^2$ in vacuum, but is subject to drag in the real world. The drag model of a vehicle is usually modeled as a force against the direction of the vehicle’s motion, which is quadratic in the size of the velocity [37] p. 164. Thus, each vehicle has the following model:

$$\Sigma_i: \dot{x}_i = -C_i x_i |x_i| + u_i, \quad y_i = x_i, \quad(16)$$

where $x_i$ is the velocity of the $i$-th vehicle, and $C_i$ is a constant which aggregates different parameters affecting the drag, e.g. density of the air, viscosity of the air, the geometry of the vehicle, and the mass of the vehicle. The vehicles are trying to coordinate their velocity - Agents #1–#7 want to travel at 60 km/h, agents #8–#13 want to travel at 70 km/h, and agents #14–#20 want to travel at 50 km/h. The edge controllers are static nonlinearities given by sigmoid functions of the form $\mu_e = \tanh(\zeta_e)$. This diffusively coupled network satisfies both Assumptions 1 and 3 and we note both the agents and the controllers are nonlinear. We choose an interaction graph $G$ as seen in Figure 2. One can check that $G$ is 4-connected, either using Menger’s theorem or using other known algorithms [38].

The parameters $C_i$ were chosen as log-uniformly between 0.01 and 0.1. The initial velocity of the agents was chosen to be Gaussian with mean $\mu = 70$ km/h and standard deviation $\sigma = 20$ km/h. We solve the synthesis problem, forcing the network to converge to $y^* = \{60 \cdot \mathbb{1}_3^\top, 70 \cdot \mathbb{1}_6^\top, 50 \cdot \mathbb{1}_5^\top\}$, where $\mathbb{1}_m$ is the all-one vector of size $m$, allowing up to 2 edges to fault. We run our FDI protocol, implementing the profile-based convergence assertion scheme, sampling the system at 10 Hz (i.e., a modified version of Algorithm 7). We consider four different scenarios, each lasting 100 seconds.

1) A faultless scenario.
2) At time $t = 20_{sec}$, the edge \{2,3\} faults, and at time $t = 50_{sec}$, the edge \{13,14\} faults.
3) At time $t = 20_{sec}$, the edge \{2,3\} faults, and at time $t = 21_{sec}$, the edge \{13,14\} faults.
4) At time $t = 0.5_{sec}$, the edge \{2,3\} faults, and at time $t = 4_{sec}$, the edge \{13,14\} faults.

The first scenario tests the nominal behavior of the protocol. The second tests its ability to handle with one single fault at a time. The third tests its ability to handle more than one fault at a time. The last tests its ability to deal with faults before the network converged. The results are available in Figures 3, 4, 5, and 6. It can be seen that we achieve the control goal in all four scenarios. Moreover, in all scenarios and at all times, the velocities of the agents are not too far from the values found in $y^*$, meaning that this protocol cannot harm the agents by demanding them to have very wild states. In the second and third scenario, the exploratory phases begins at the first measurement after the fault occurred. On the contrary, in the fourth scenario, it takes the exploratory phase begins only at $t = 1.3_{sec}$, a little under a second after the first fault. This is because the steady-states of the faulty and nominal closed-loop system are relatively close, meaning it takes a little extra time to find that a fault exists.

VIII. CONCLUSION

We considered multi-agent networks prone to communication faults in which the agents are output-strictly MEIP
and the controllers are MEIP. We exhibited a protocol in which a constant bias $w$ is added to the controller output, and showed that if $w$ is chosen randomly, no matter what the underlying graph $G$ is, we can asymptotically differentiate between any two versions (faulty or faultless) of the system. We also showed that if $w$ is chosen randomly within a certain set, we asymptotically differentiate the faultless version of the system from its faulty version, while also solving the synthesis problem for the faultless version, assuming $G$ was connected enough, i.e., 2-connected. These results were used to describe algorithms for network fault detection and isolation protocols for general MEIP multi-agent systems, where the number of isolable faults is given by a graph-theoretic characteristic of $G$, while no extra information on the agents and controllers but MEIP is used. These were achieved by assuming the existence of an on-line algorithm asserting that a given network converges to a conjectured steady-state, allowing us to move from asymptotic differentiation to on-line differentiation. Later, two such algorithms were built using passivity of the agents and controllers while no extra information on the underlying graph.

REFERENCES

[1] K.-K. Oh, M.-C. Park, and H.-S. Ahn, “A survey of multi-agent formation control,” Automatica, vol. 53, pp. 424 – 440, 2015.
[2] L. Scardovi, M. Arcak, and E. D. Sontag, “Synchronization of interconnected systems with applications to biochemical networks: An input-output approach,” IEEE Transactions on Automatic Control, vol. 55, no. 6, pp. 1367–1379, 2010.
[3] R. Olfati-Saber, J. A. Fax, and R. M. Murray, “Consensus and cooperation in networked multi-agent systems,” Proceedings of the IEEE, vol. 95, pp. 215–233, Jan. 2007.
[4] A. Teixeira, I. Shames, H. Sandberg, and K. H. Johansson, “Distributed fault detection and isolation resilient to network model uncertainties,” IEEE Transactions on Cybernetics, vol. 44, pp. 2024–2037, Nov 2014.
[5] S. Jafari, A. Ajorlou, A. G. Aghdam, and S. Tafazoli, “On the structural controllability of multi-agent systems subject to failure: A graph-theoretic approach,” in 49th IEEE Conference on Decision and Control (CDC), pp. 4565–4570, 2010.
[6] S. Jafari, A. Ajorlou, and A. G. Aghdam, “Leader localization in multi-agent systems subject to failure: A graph-theoretic approach,” Automatica, vol. 47, no. 8, pp. 1744–1750, 2011.
[7] M. A. Rahimian and V. M. Preciado, “Detection and isolation of failures in directed networks of LTI systems,” IEEE Transactions on Control of Network Systems, vol. 2, no. 2, pp. 183–192, 2014.
[8] M. A. Rahimian and V. M. Preciado, “Failure detection and isolation in integrator networks,” in 2015 American Control Conference (ACC), pp. 677–682, 2015.
[9] G. Battistelli and P. Tesi, “Detecting topology variations in dynamical networks,” in 2015 54th IEEE Conference on Decision and Control (CDC), pp. 3349–3354, IEEE, 2015.
APPENDIX

This appendix includes the proof of various technical propositions from Subsection VI-B. We start with the proof of Proposition 6.

Proof. Without loss of generality, we assume that \( \Omega = \Omega^* \) for all \( i \), as proving that \( \sum_{i=1}^{n} \rho_i \Omega_i(\theta_i) \geq C \Omega^*(\sum_{i=1}^{n} \theta_i) \) would imply the desired inequality. We also assume that \( \rho_i = 1 \) for all \( i \), as proving that \( \sum_{i} \Omega_i(\theta_i) \geq C \Omega^*(\sum_{i=1}^{n} \theta_i) \) would give \( \sum_{i} \rho_i \Omega_i(\theta_i) \geq C \min_{i} \rho_i \cdot \Omega^*(\sum_{i=1}^{n} \theta_i) \). Define \( F : [0, D]^{n} \setminus \{0\} \to \mathbb{R} \) as \( F(\theta_1, \ldots, \theta_n) = \frac{\sum_{i=1}^{n} \Omega_i(\theta_i)}{\Omega^*(\sum_{i=1}^{n} \theta_i)} \), where the claim is equivalent to \( F \) being bounded from below. For any \( r > 0 \), \( F \) is continuous on the compact set \([0, D]^n \setminus \{x : \|x\| > r\}\), so its minimum is obtained at some point. As \( F \) does not vanish on the set, the minimum is positive, so \( F \) is bounded from below on that set by a constant greater than zero. It remains to show that \( \lim n \inf_{\theta_1, \ldots, \theta_n} F(\theta_1, \ldots, \theta_n) > 0 \). Let \( \beta = \max_i \beta_i \), so that \( \lim n \inf_{\theta_1, \ldots, \theta_n} \frac{\Omega_i(\theta_i)}{\Omega^*(\sum_{i=1}^{n} \theta_i)} > 0 \). Then \( F(\theta_1, \ldots, \theta_n) = \frac{\sum_{i=1}^{n} \Omega_i(\theta_i)}{\Omega^*(\sum_{i=1}^{n} \theta_i)} \). We want to bound both factors from below when \( \theta_1, \ldots, \theta_n \to 0 \). It is clear that the second factor is equal to \( (\lim n \inf_{\theta_1, \ldots, \theta_n} \frac{\Omega_i(\theta_i)}{\Omega^*(\sum_{i=1}^{n} \theta_i)})^{-1} \), which is a positive real number by assumption. As for the first factor, we can bound it as \( \lim n \inf_{\theta_1, \ldots, \theta_n} \frac{\sum_{i=1}^{n} \Omega_i(\theta_i)}{\Omega^*(\sum_{i=1}^{n} \theta_i)} \geq \lim n \inf_{\theta_1, \ldots, \theta_n} \frac{\sum_{i=1}^{n} \Omega_i(\theta_i)}{\Omega^*(\max_i \theta_i)} \geq 0 \) as \( \sum_{i=1}^{n} \theta_i \leq n \max_i \theta_i \) and \( \sum_{i=1}^{n} \Omega_i(\theta_i) \geq \Omega^*(\max_i \theta_i) \). This completes the proof. \( \square \)

We now prove Proposition 7.

Proof. By assumption \( \hat{S}(t_k) \) is monotone decreasing and bounded from below, as \( \hat{S}(t_k) \geq 0 \). Thus it converges to some value, denoted \( \hat{S}_\infty \). Using \( \hat{S}(t_{k+1}) - \hat{S}(t_k) \leq -\Omega(\hat{S}(t_{k+1}))(t_{k+1} - t_k) \) and taking \( k \to \infty \) gives that \( 0 \leq -\Omega(\hat{S}_\infty) \). However, \( \Omega(\cdot) \) is non-negative, so we must have \( \hat{S}_\infty = 0 \), and thus \( S_\infty = 0 \), meaning that \( \hat{S}(t_k) \to 0 \). By monotonicity of \( \hat{S} \), we conclude that \( \hat{S}(t) \to 0 \) as \( t \to \infty \). \( \square \)