
Discrete Event Dynamic Systems: Theory and Applications, 10, 33–86 (2000)

Coordinated Decentralized Protocols for Failure Diagnosis of Discrete Event Systems

RAMI DEBOUK ridebouk@eecs.umich.edu
Department of Electrical Engineering and Computer Science, The University of Michigan, 1301 Beal Avenue, Ann Arbor, MI 48109–2122, USA

STÉPHANE LAFORTUNE stephane@eecs.umich.edu
Department of Electrical Engineering and Computer Science, The University of Michigan, 1301 Beal Avenue, Ann Arbor, MI 48109–2122, USA

DEMOSTHENIS TENEKETZIS teneket@eecs.umich.edu
Department of Electrical Engineering and Computer Science, The University of Michigan, 1301 Beal Avenue, Ann Arbor, MI 48109–2122, USA

Abstract. We address the problem of failure diagnosis in discrete event systems with decentralized information. We propose a coordinated decentralized architecture consisting of local sites communicating with a coordinator that is responsible for diagnosing the failures occurring in the system. We extend the notion of diagnosability, originally introduced in Sampath et al. (1995) for centralized systems, to the proposed coordinated decentralized architecture. We specify three protocols that realize the proposed architecture; each protocol is defined by the diagnostic information generated at the local sites, the communication rules used by the local sites, and the coordinator's decision rule. We analyze the diagnostic properties of each protocol. We also state and prove conditions for a language to be diagnosable under each protocol. These conditions are checkable off-line. The on-line diagnostic process is carried out using the diagnosers introduced in Sampath et al. (1995) or a slight variation of these diagnosers. The key features of the proposed protocols are: (i) they achieve, each under a set of assumptions, the same diagnostic performance as the centralized diagnoser; and (ii) they highlight the "performance vs. complexity" tradeoff that arises in coordinated decentralized architectures. The correctness of two of the protocols relies on some stringent global ordering assumptions on message reception at the coordinator's site, the relaxation of which is briefly discussed.

Keywords: failure diagnosis, decentralized information, diagnostic protocols

1. Introduction

Failure detection and isolation is an important task in the automatic control of large complex systems. In order to guarantee a reliable system performance, the control engineer should guarantee that the system is running safely within its normal boundaries. Consequently, the problem of failure diagnosis has received considerable attention in the literature. Many schemes ranging from fault-tree (Lapp and Powers, 1977) and analytical redundancy (Willsky, 1976; Frank, 1990) methods to discrete event system (DES) approaches (Sampath et al., 1995; Lin, 1994; Bavishi and Chong, 1994; Holloway and Chand, 1994; Boubour et al., 1997; Cassandras and Lafortune, 1998), model based reasoning (Davis and Hamscher, 1992) and expert systems (Scherer and White, 1987) methods, have been proposed to approach this problem. For a brief description of these methods and additional references,

the interested reader is referred to Pouliezos and Stavrakakis (1994) and the introduction of Sampath et al. (1995).

Almost all of the abovementioned approaches have been developed for systems where the information used for fault diagnosis is centralized. A notable exception is Holloway and Chand (1994), where the authors present a distributed fault monitoring method, time templates. Time templates monitoring is cited to have the advantage of being easily implemented in distributed control architectures. Many systems are decentralized in nature, for instance, the majority of technological complex systems (computer and communication networks, manufacturing, process control and power systems, etc.) are informationally decentralized. In decentralized information systems there are several work stations (decision makers, controllers, diagnosers) each having access to its own local information. The stations may communicate and exchange limited information among each other. Since this information is exchanged in real-time and over channels of limited capacity, there are propagation delays, along with faults and transmission errors. Thus, the information available to each station is incomplete, delayed, and possibly erroneous. Hence, the approaches to failure diagnosis mentioned above do not apply directly to informationally decentralized systems. Consequently, it is important to develop diagnostic methodologies for informationally decentralized systems. This fact is also recognized in Holloway and Chand (1994) and Boubour et al. (1997).

In this paper, we investigate failure diagnosis problems in DES under decentralized information. Having adopted a DES approach to failure diagnosis, we extend the notion of diagnosability, introduced in Sampath et al. (1995) for centralized systems, to a coordinated decentralized architecture consisting of local sites communicating with a coordinator that is responsible for diagnosing the failures occurring in the system. We present three specific protocols that realize the architecture under consideration. A protocol specifies the diagnostic information generated at each local site, the communication rules used by the local sites, and the decision rule for failure diagnosis employed by the coordinator. We present and discuss the diagnostic properties of the suggested protocols. We state and prove conditions for a language to be diagnosable under these protocols and provide off-line tests to check the diagnosability property. The on-line diagnostic process is carried out by the diagnosers introduced in Sampath et al. (1995) or a slight variation of these diagnosers. The key features of the coordinated decentralized protocols presented in this paper are: first, they perform as well as the centralized diagnoser each under a set of assumptions; and second, they highlight the "performance vs. complexity" tradeoff that arises in coordinated decentralized architectures. The correctness of two of the protocols relies on some stringent global ordering assumptions on message reception at the coordinator's site, the relaxation of which is briefly discussed.

This paper is organized as follows. In Section 2, we present some preliminary definitions and results that are critical for the development of the technical results in this paper. We provide an overview of the coordinated decentralized architecture under consideration in Section 3. We specify three protocols that realize this architecture in Sections 4, 5, and 6. We describe each protocol in detail; that is, we precisely specify the diagnostic information generated at local sites, the communication rules used between the local sites and

the coordinator and the coordinator's decision rule for failure diagnosis. We analyze the diagnostic properties of each protocol, and discover conditions to ensure diagnosability of a language under each protocol. We present and discuss the performance vs. complexity tradeoff highlighted by the three protocols and the relaxation of the ordering assumption in Section 7. We draw some conclusions and discuss the contribution of the paper in Section 8.

2. Preliminaries

2.1. The System Model

The system to be diagnosed is modeled as a FSM

$$G = (X, \Sigma, \delta, x_0) \tag{1}$$

where X is the state space, Σ is the set of events, δ is the partial transition function, and x_0 is the initial state of the system. The model G accounts for the normal and failed behavior of the system. The behavior of the system is described by the prefix-closed language (Ramadge and Wonham, 1989) $L(G)$ generated by G. $L(G)$ is a subset of Σ^*, where Σ^* denotes the Kleene closure of the set Σ (Hopcroft and Ullman, 1979). In this paper we will use the language $L(G)$, or simply L, and the system interchangeably.

Some of the events in Σ are observable, i.e., their occurrence can be observed, while the rest are unobservable. Thus, the event set Σ is partitioned as $\Sigma = \Sigma_o \cup \Sigma_{uo}$ where Σ_o represents the set of observable events and Σ_{uo} the set of unobservable events. The observable events in the system may be one of the following: commands issued by the controller, sensor readings occurring after the execution of those above commands, and changes in sensor readings. The unobservable events may be failure events or other events that cause changes in the system state not recorded by sensors (see Sampath, 1995; Sampath et al., 1996).

Let $\Sigma_f \subseteq \Sigma$ denote the set of failure events which are to be diagnosed. We assume, without loss of generality, that $\Sigma_f \subseteq \Sigma_{uo}$, since an observable failure event can be trivially diagnosed. Our objective is to identify the occurrence, if any, of the failure events, given that in the traces generated by the system, only the events in Σ_o are observed. In this regard, we partition the set of failure events into disjoint nonempty sets corresponding to different failure types

$$\Sigma_f = \Sigma_{f1} \cup \Sigma_{f2} \cup \cdots \cup \Sigma_{fm}. \tag{2}$$

Let Π_f denote this partition. For the motivation of such a partition, the reader is referred to Sampath et al. (1995) and Sampath et al. (1996). Hereafter, when we write a failure of type F_i has occurred, we will mean that some event of the set Σ_{fi} has occurred.

2.2. Notation

The empty trace is denoted by ϵ. Let $\overline{s}$ denote the prefix-closure of any trace $s \in \Sigma^*$. We define $\|s\|$ to be the length of trace s. Whenever we say that there exists a trace s of *arbitrarily long length* having a given property, we mean the following: for all integers n, there exists s, such that $\|s\| > n$ and s possesses the given property. We denote by L/s the post-language of L after s, i.e.,

$$L/s = \{t \in \Sigma^* \mid st \in L\}. \tag{3}$$

We define the projection $P\colon \Sigma^* \to \Sigma_o^*$ in the usual manner (Ramadge and Wonham, 1989)

$$\begin{aligned} P(\epsilon) &= \epsilon, \\ P(\sigma) &= \sigma \text{ if } \sigma \in \Sigma_o, \\ P(\sigma) &= \epsilon \text{ if } \sigma \in \Sigma_{uo}, \\ P(s\sigma) &= P(s)P(\sigma),\ s \in \Sigma^*,\ \sigma \in \Sigma. \end{aligned} \tag{4}$$

The inverse projection operator P_L^{-1} is defined as

$$P_L^{-1}(y) = \{s \in L\colon P(s) = y\}. \tag{5}$$

Let s_f denote the final event of trace s. We define

$$\Psi(\Sigma_{fi}) = \{s\sigma_f \in L\colon \sigma_f \in \Sigma_{fi}\}, \tag{6}$$

i.e., $\Psi(\Sigma_{fi})$ denotes the set of all traces that end in a failure event belonging to the class Σ_{fi}. Consider $\sigma \in \Sigma$ and $s \in \Sigma^*$. We use the notation $\sigma \in s$ to denote that σ is an event in the trace s. With slight abuse of notation, we write $\Sigma_{fi} \in s$ to denote the fact that $\sigma_f \in s$ for some $\sigma_f \in \Sigma_{fi}$, or formally, $\overline{s} \cap \Psi(\Sigma_{fi}) \neq \emptyset$. We also define

$$X_o = \{x_0\} \cup \{x \in X\colon x \text{ has an observable event into it}\}. \tag{7}$$

Let $L(G, x)$ denote the set of all traces that originate from state x of G. We define

$$L_o(G, x) = \{s \in L(G, x)\colon s = u\sigma, u \in \Sigma_{uo}^*, \sigma \in \Sigma_o\} \tag{8}$$

and

$$L_\sigma(G, x) = \{s \in L_o(G, x)\colon s_f = \sigma\}. \tag{9}$$

$L_o(G, x)$ denotes the set of all traces that originate from state x and end at the first observable event, while $L_\sigma(G, x)$ denotes those traces in $L_o(G, x)$ that end with the particular observable event σ.

The generator G' (see Sampath et al., 1995; Sampath, 1995) is the nondeterministic FSM,

$$G' = (X_o, \Sigma_o, \delta_{G'}, x_0), \tag{10}$$

where X_o, Σ_o, and x_0 are defined as previously, and the transition relation of G' is given by $\delta_{G'} \subseteq (X_o \times \Sigma \times X_o)$ and is defined as follows:

$$(x, \sigma, x') \in \delta_{G'} \text{ if } \delta(x, s) = x' \text{ for some } s \in L_\sigma(G, x). \tag{11}$$

It is easy to verify that $L(G') = P(L)$ where

$$P(L) = \{t\colon\ t = P(s) \text{ for some } s \in L\}. \tag{12}$$

2.3. Definition of Diagnosability

Loosely speaking, a language is said to be diagnosable with respect to a set of observable events and a failure partition if within a finite delay, the occurrence of any failure can be detected using the history of observable events. More rigorously, diagnosability is defined as follows (Sampath et al., 1995; Sampath, 1995):

Definition 1. A prefix-closed and live language L is said to be diagnosable with respect to the projection P and with respect to the partition Π_f on Σ_f if the following holds

$$(\forall i \in \Pi_f)(\exists n_i \in \mathbb{N})(\forall s \in \Psi(\Sigma_{fi}))(\forall t \in L/s)(\|t\| \geq n_i \Rightarrow D)$$

where the diagnosability condition D is

$$(\forall w \in P_L^{-1}(P(st)))\ (\Sigma_{fi} \in w).$$

(A language L is live if for all $s \in L$, there exists $\sigma \in \Sigma$ such that $s\sigma \in L$.) Note here that the above definition is only applicable to centralized systems, since it assumes the availability of all the system information at one (centralized) center or site: there is only one projection P that observes the behavior of the system, in addition to a single inverse projection P_L^{-1}, and both are used to check the diagnosability condition D.

2.4. The Diagnoser

The diagnoser is a FSM built from the system model G. This machine is used to perform diagnostic when it observes on-line the behavior of the system. We first define the set of failure labels $\Delta_f = \{F_1, F_2, \ldots, F_m\}$ where $|\Pi_f| = m$, and the complete set of possible labels

$$\Delta = \{N\} \cup 2^{\Delta_f}. \tag{13}$$

Here N is to be interpreted as meaning normal, while F_i, $i \in \{1, 2, \ldots, j\}$ as meaning that a failure of type F_i has occurred. Recall, from Equation 7, the definition of X_o and define

$$Q_o = 2^{X_o \times \Delta}. \tag{14}$$

The diagnoser for G is the FSM

$$G_d = (Q_d, \Sigma_o, \delta_d, q_0) \tag{15}$$

where Q_d, Σ_o, δ_d, and q_0 have the usual interpretation of state space, event set, transition function, and initial state. The initial state of the diagnoser is defined to be $\{(x_0, \{N\})\}$. The transition function δ_d of the diagnoser is constructed in a similar manner to the transition function of an observer of G (Hopcroft and Ullman, 1979), with an additional aspect that includes attaching failure labels to the states and propagating these labels from state to state. For more information about the construction of the diagnoser, the reader is referred to Sampath et al. (1995) and Sampath (1995). The state space Q_d is the resulting subset of Q_o composed of the states of the diagnoser that are reachable from q_0 under δ_d. Since the state space Q_d of the diagnoser is a subset of Q_o, a state q_d of G_d is of the form $q_d = \{(x_1, l_1), \ldots, (x_n, l_n)\}$, where $x_i \in X_o$ and $l_i \in \Delta$.

Next, we provide some definitions that are necessary in order to state the main diagnosability result for centralized systems in Section 2.5. For a detailed discussion and interpretation of this material the reader is referred to Sampath et al. (1995) and Sampath (1995).

Definition 2. (Definition 6-1 in Sampath et al., 1995). A state $q \in Q_d$ is said to be F_i-certain if $\forall (x, l) \in q$, $F_i \in l$.

Definition 3. (Definition 6-3 in Sampath et al., 1995). A state $q \in Q_d$ is said to be F_i-uncertain if $\exists (x, l), (y, l') \in q$, x not necessarily distinct from y, such that $F_i \in l$ and $F_i \notin l'$.

Definition 4. (Definition 7 in Sampath et al., 1995). A set of states $x_1, x_2, \ldots, x_n \in X$ is said to form a cycle in G if $\exists s \in L(G, x_1)$ such that $s = \sigma_1 \sigma_2 \ldots \sigma_n$, and $\delta(x_l, \sigma_l) = x_{(l+1) \bmod n}$, $l = 1, 2, \ldots, n$.

Definition 5. (Definition 8 in Sampath et al., 1995). A set of F_i-uncertain states $q_1, q_2, \ldots, q_n \in Q_d$ is said to form an F_i-indeterminate cycle if

1) $q_1, q_2, \ldots, q_n$ form a cycle in G_d with $\delta_d(q_l, \sigma_l) = q_{l+1}, l = 1, 2, \ldots, n-1, \delta_d(q_n, \sigma_n) = q_1, \sigma_l \in \Sigma_o, l = 1, 2, \ldots, n$.

2) $\exists (x_l^k, l_l^k), (y_l^r, \tilde{l}_l^r) \in q_l$, x_l^k not necessarily distinct from y_l^r, $l = 1, 2, \ldots, n$, $k = 1, 2, \ldots, m$, and $r = 1, 2, \ldots, m'$ such that

   a) $F_i \in l_l^k$, $F_i \notin \tilde{l}_l^r$ for all l, k, and r.

   b) The sequence of states $\{x_l^k\}$, $l = 1, 2, \ldots, n$, $k = 1, 2, \ldots, m$, and $\{y_l^r\}$, $l = 1, 2, \ldots, n$, $r = 1, 2, \ldots, m'$ form cycles in G' with

$$(x_l^k, \sigma_l, x_{l+1}^k) \in \delta_{G'}, \quad l = 1, 2, \ldots, n-1, \quad k = 1, 2, \ldots, m,$$
$$(x_n^k, \sigma_n, x_1^{k+1}) \in \delta_{G'}, \quad k = 1, 2, \ldots, m-1, \text{ and } (x_n^m, \sigma_n, x_1^1) \in \delta_{G'}.$$

and

$$(y_l^r, \sigma_l, y_{l+1}^r) \in \delta_{G'}, \quad l = 1, 2, \ldots, n-1, \quad r = 1, 2, \ldots, m',$$
$$(y_n^r, \sigma_n, y_1^{r+1}) \in \delta_{G'}, \quad r = 1, 2, \ldots, m'-1, \text{ and } (y_n^{m'}, \sigma_n, y_1^1) \in \delta_{G'}.$$

An F_i-indeterminate cycle in G_d indicates the presence in L of two traces s_1 and s_2 of arbitrarily long length, such that they both have the same observable projection and s_1 contains a failure event from the set Σ_{fi} while s_2 does not.

Finally, the following lemma relates the properties of a diagnoser state to the properties of the traces in the language.

LEMMA 1 (Lemma 2 in Sampath et al., 1995)

i) Let $\delta_d(q_0, u) = q$. If q is F_i-certain, then $\forall w \in P_L^{-1}(u)$, $\Sigma_{fi} \in w$.

ii) If a state $q \in Q_d$ is F_i-uncertain, then this implies that $\exists s_1, s_2 \in L$ such that $\Sigma_{fi} \in s_1$, $\Sigma_{fi} \notin s_2$, $P(s_1) = P(s_2)$, and $\delta_d[q_0, P(s_1)] = q$.

We note here that all of the notation introduced in this subsection and the previous ones assumes that the set of observable events is Σ_o. Later on, we will be using subsets of Σ_o, namely Σ_{o1} and Σ_{o2}; the above notation will still be applicable to the subsets of Σ_o, with the minor change, when necessary, of adding subscripts: a "1" subscript will be used in notation related to Σ_{o1}, while a "2" subscript will be used in notation related to Σ_{o2}. In this case, we define Σ_o to be $\Sigma_{o1} \cup \Sigma_{o2}$.

2.5. Necessary and Sufficient Conditions for Diagnosability

It is intuitive, based on the definition of diagnosability, the properties of the diagnoser, and Definition 5, that in order for a language to be diagnosable, the diagnoser should not have any F_i-indeterminate cycles for all failure types F_i. This condition is stated formally as follows:

THEOREM 1 (Theorem 2 in Sampath et al., 1995) *A language L is diagnosable with respect to the projection P and the failure partition Π_f on Σ_f if and only if its diagnoser G_d satisfies the following condition: there are no F_i-indeterminate cycles in G_d for all failure types F_i.*

3. General Specification of the Problem

3.1. A Coordinated Decentralized Architecture

In decentralized systems, the global system information is distributed at several sites. The "agents" at different sites may communicate and exchange information in real time, or just

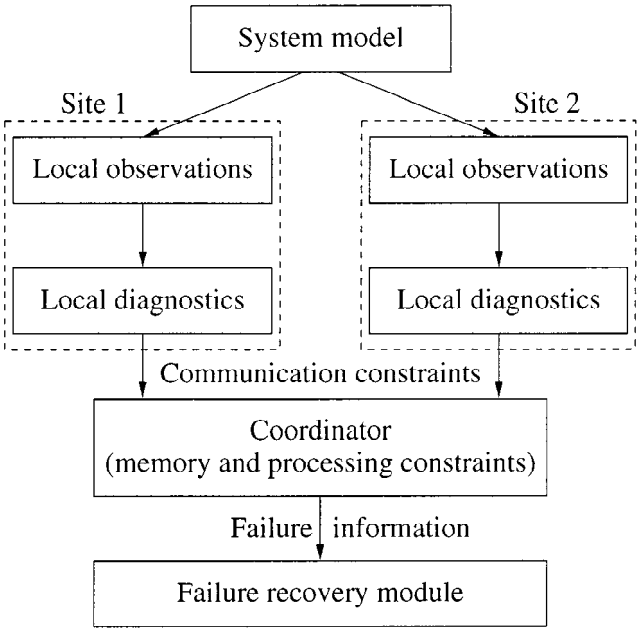

Figure 1. Coordinated decentralized architecture.

report some or all of their information to a center that, in general, possesses limited knowledge about the system. For each distinct information flow we obtain a distinct decentralized architecture. In this paper, we restrict attention to a coordinated decentralized architecture with two local sites communicating with a coordinator. This architecture is depicted in Figure 1. In this section, we discuss this architecture. We present protocols that realize the architecture in Sections 4, 5, and 6.

In Figure 1, the top block represents the system model, or G in the notation of Section 2.1. G models the synchronization of the interaction of all the components that constitute the system (see Sampath, 1995; Sampath et al., 1996). Each site is composed of two modules: an observation module and a diagnostic module. The site i, $i \in \{1, 2\}$, locally observes the system based on its available sensing capabilities. Therefore, a projection P_i is associated with site i, where P_i is defined on the set of observable events Σ_{oi} (note here that Σ_{o1} and Σ_{o2} need not be disjoint although sites 1 and 2 may be physically apart). The union of Σ_{o1} and Σ_{o2} is the set of observable events Σ_o. Site i locally processes its own observation and generates its diagnostic information. Both sites communicate some form of their diagnostic information to the coordinator. The type of information communicated is determined by the communication rules used by the sites. The task of the coordinator is to process, according to a prescribed decision rule, the messages received from both sites to infer occurrences of failures. If a failure is detected by the coordinator, it is broadcast to the failure recovery module.

We intend to investigate diagnosability properties of the above architecture under the following assumptions.

A1 $L(G)$ is live.

A2 G has no cycles of unobservable events with respect to either Σ_{o1} or Σ_{o2}.

A3 $L(G)$ is not diagnosable with respect to P_i and Π_f on Σ_f, $i = 1, 2$.

A4 There is reliable communication between the local sites and the coordinator, i.e., all messages sent from a local site are received by the coordinator correctly and in order.

A5 Messages communicated between the local sites and the coordinator are received in the order they are sent (globally).

A6 The sets of observable events at each site are common knowledge (Aumann, 1976; Washburn and Teneketzis, 1984) to all sites.

A7 The two sites are allowed to report to the coordinator only some processed version of their raw data.

A8 The coordinator does not have a model of the system, that is, it does not know the dynamics of the system. It has a simple structure; specifically, it has limited memory and limited processing capabilities.

Assumption **A1** ensures that there are no deadlocks. This assumption can be relaxed easily as discussed in Sampath (1995) and Sampath et al. (1998). Assumption **A2** ensures that observations occur with some regularity with respect to both P_1 and P_2: since detection of failures is based on observable transitions of the system, we require that G does not generate arbitrarily long sequences of unobservable events with respect to either P_1 or P_2. Assumption **A3** eliminates the trivial case where even though the observable events are partitioned, the system is still diagnosable (in a centralized setup) with respect to one of the projections and the failure partition. In such a case the decentralized architecture is necessarily diagnosable! Assumption **A5** ensures that the global order of all messages received by the coordinator is preserved. Assumptions **A4**, **A6**, and **A7** are self explanatory. Finally, Assumption **A8** is consistent with features of hierarchical organizations. Assumptions **A1**–**A8** will be used, even if not explicitly stated, in the derivation of all the results of this paper, unless otherwise specified.

3.2. Definition of Diagnosability

As noted in Section 2.3, the definition of diagnosability in Sampath et al. (1995) (Definition 1 in this paper) assumes centralization of the available information; hence it is not directly applicable to coordinated decentralized systems. Moreover, the coordinated decentralized architecture in Figure 1 represents a class of realizations of the same architecture where the choice of local diagnostic rules, communication rules, and decision rules, defines one realization. Therefore, to define diagnosability for coordinated decentralized systems, we

need to account for the rules used to generate local diagnostic information together with the associated communication rules and the coordinator's decision rule for failure diagnosis. In the proposed coordinated architecture the local agents do not interact with one another; they only communicate with the coordinator that is assigned the task of detecting and isolating failures. Let C denote the coordinator's diagnostic information. For each sample path of the DES, C is represented by an information set that is protocol-dependent. For instance in Protocol 3 (cf. Section 6.2.3), C is described by a set of failure labels; in Protocol 2 (cf. Section 5.2.3), C is described by a diagnoser state. The description of C in the case of Protocol 1 is more complex and is presented in Section 4.1.3.

Definition 6. The coordinator's diagnostic information C is said to be F_i-certain if based on C, the coordinator is certain that a failure of type F_i has occurred.

We mentioned earlier that a protocol realizes one instance of the coordinated decentralized architecture of Figure 1. We formalize this notion of protocol as follows:

Definition 7. Within the context of the coordinated decentralized architecture described in Section 3.1 and depicted in Figure 1, a protocol is defined by the diagnostic information generated at the local sites, the rules used by the local sites to communicate to the coordinator, and the decision rule used at the coordinator site.

Using Definitions 6 and 7 we can define diagnosability under a given protocol.

Definition 8. A prefix-closed and live language L is said to be diagnosable under a protocol, a set of projections P_1, P_2 and a failure partition Π_f on Σ_f if the following holds

$$(\forall i \in \Pi_f)(\exists n_i \in \mathbb{N})(\forall s \in \Psi(\Sigma_{fi}))(\forall t \in L/s)(\|t\| \geq n_i \Rightarrow C \text{ is } F_i\text{-certain}).$$

Thus diagnosability, as defined above, requires that the detection of any failure should be achieved by the coordinator within a finite delay of the occurrence of that failure.

3.3. Objective

Any realization of the coordinated decentralized architecture of Section 3.1 cannot outperform the centralized one. Hence, a desirable objective in realizing such an architecture is to aim at diagnosing all failure types that can be diagnosed by the centralized diagnoser. Therefore, the design process should determine a failure diagnosis protocol that performs as well as the centralized diagnoser would. In case this is not feasible, conditions on the system structure may be found to guarantee that the protocol diagnoses all failure types that are diagnosed by the centralized diagnoser. Note here that according to Definition 8, the set of projections and the failure partition are given and fixed; more generally, they could be included in the protocol. The next three sections describe three protocols that achieve the above objective.

4. A Coordinated Decentralized Protocol: Protocol 1

4.1. Specification of the Protocol

In this section, we present a protocol for the preceding coordinated decentralized architecture that is capable of diagnosing the same types of failures as the ones diagnosed using a centralized diagnoser. The specification of the protocol is done under Assumptions **A1**–**A8** of Section 3.1. Thereafter, we will refer to this protocol as Protocol 1. We begin by specifying the type of diagnostic information generated at local sites.

4.1.1. Diagnostic Information at Local Sites

The diagnostic information at the local site is generated by the extended diagnoser defined below. The *extended diagnoser* for G was first introduced in Sampath (1993), and it is the FSM

$$G_d^e = \left(Q_d^e, \Sigma_o, \delta_d^e, q_0^e\right) \tag{16}$$

where Q_d^e, Σ_o, δ_d^e, and q_0^e have the usual interpretation of state space, event set, transition function, and initial state. The initial state of the extended diagnoser is defined to be $\{(x_0, \{N\}), (x_0, \{N\})\}$. A state $q \in Q_d^e$ is of the form

$$q = \{((x_1, l_1), (x_1', l_1')), ((x_2, l_2), (x_2', l_2')), ((x_2, l_2), (x_2'', l_2'')), \ldots, ((x_n, l_n), (x_n', l_n'))\}$$

where each (x, l) pair is in Q_o, i.e., $x \in X_o$ and $l \in \Delta$. A tuple of (x, l) pairs, say $((x_1, l_1), (x_1', l_1'))$, has the following meaning: x_1' is a component of a system state estimate after the occurrence of an observable event and l_1' is its failure label, while x_1 is the immediate predecessor state of x_1' in G' and l_1 is its corresponding failure label. The transition function δ_d^e of the extended diagnoser is constructed in a manner similar to the transition function of the diagnoser G_d, with the additional aspect that every state of G that appears in a state component of G_d is associated with its immediate predecessor state in G' (along the sub-trace of events under consideration) and both states carry their labels; these labels are attained following the same label propagation rules as in Sampath et al. (1995). The state space Q_d^e is the resulting subset of $Q_o \times Q_o$ composed of the states of the extended diagnoser that are reachable from q_0^e under δ_d^e. By construction, $L(G_d^e) = L(G_d) = P(L)$. We illustrate the construction of extended diagnosers in the following example.

Example 1. Consider the system shown in Figure 2 with $\Sigma = \{a, b, c, d, e, \sigma\}$, $\Sigma_{uo} = \{\sigma\}$, $\Sigma_{f1} = \{\sigma\}$, $\Sigma_{o1} = \{a, c, d, e\}$, and $\Sigma_{o2} = \{b, d, e\}$. The extended diagnosers G_{d1}^e and G_{d2}^e for this system are shown in Figure 3. Consider the state $q = \{(2N, 6N), (5N, 7N)\}$ in G_{d1}^e; q is read as follows: the system is either in state 6 with a normal label, or it is in state 7, also with a normal label; state 6 has been reached (by an observable event, possibly preceded by unobservable events) from state 2, while state 7 has been reached (by an observable event, possibly preceded by unobservable events) from state 5. Now the next observable event is d: if the system is at state 6, then it transitions into state 8, and since there are no

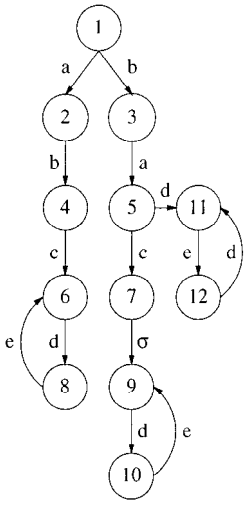

Figure 2. The system G for Example 1.

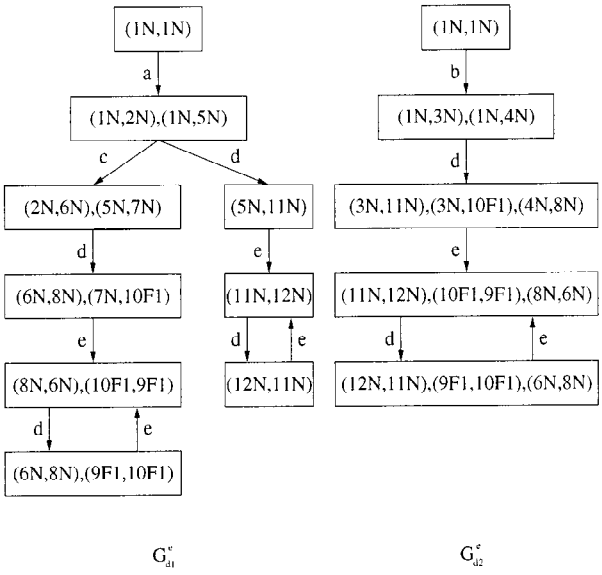

Figure 3. The extended diagnosers G^e_{d1} and G^e_{d2} for Example 1.

failure events along the path from state 6 to state 8 the resulting component of the new state estimate is $(6N, 8N)$; if the system is at state 7, it transitions into state 10 following the occurrence of the sequence σd, i.e., a failure of type F_1 has occurred along the path, and

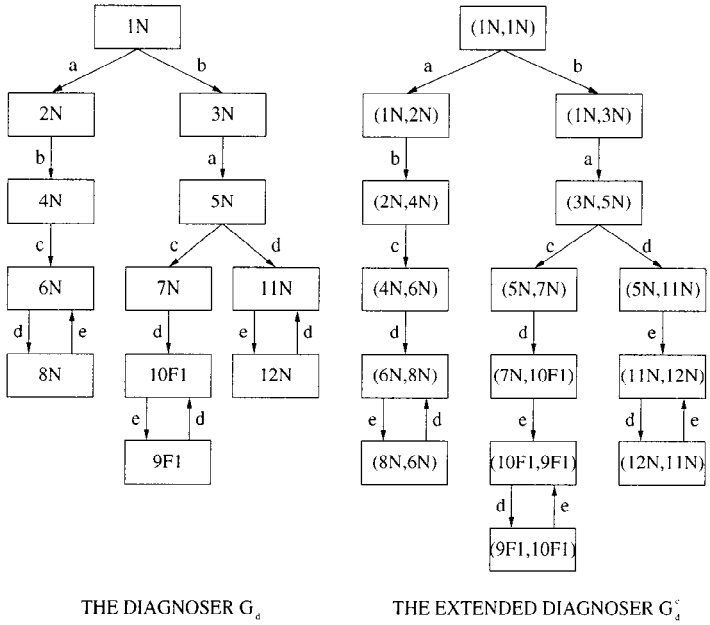

Figure 4. The diagnoser G_d and extended diagnoser G_d^e for Example 1.

the resulting other component of the new state estimate is $(7N, 10F1)$. Therefore the state of G_{d1}^e is $\{(6N, 8N), (7N, 10F1)\}$ after the occurrence of the observable event d. All other extended diagnoser states are constructed by following a similar procedure.

We define the state projection $SP\colon\ Q_o \times Q_o \to Q_o$ as follows:

$$q = \{((x_1, l_1), (x_1', l_1')), \ldots, ((x_n, l_n), (x_n', l_n'))\} \mapsto SP(q) = \{(x_1', l_1'), \ldots, (x_n', l_n')\}. \tag{17}$$

Then, with a slight abuse of notation, we have that $SP(G_d^e) = G_d$; hence, one diagnoser state may be associated with more than one extended diagnoser states. Therefore, an extended diagnoser state potentially carries more information than a diagnoser state. In the case of centralized systems, G_d and G_d^e are equivalent from the point of view of diagnosability as defined in Definition 1; it is for that reason that prior work (Sampath et al., 1995; Sampath, 1995; Sampath et al., 1996; Sampath et al., 1998) only considered the simpler G_d.

Example 2. Again consider the system shown in Figure 2 with $\Sigma = \{a, b, c, d, e, \sigma\}$, $\Sigma_{uo} = \{\sigma\}$, $\Sigma_{f1} = \{\sigma\}$. The diagnoser G_d and extended diagnoser G_d^e for this system are shown in Figure 4. We can see that the transition structure of G_d^e refines that of G_d. In particular, state $6N$ of G_d is associated with states $(4N, 6N)$ and $(8N, 6N)$ in G_d^e since $SP((4N, 6N)) = SP((8N, 6N)) = 6N$.

We define the *unobservable reach* of an extended diagnoser state as follows.

Definition 9. Let $q = \{((x_1, l_1), (x'_1, l'_1)), \ldots, ((x_n, l_n), (x'_n, l'_n))\}$ be a state of the extended diagnoser G^e_{dj}, $j \in \{1, 2\}$. Define the set

$$S_j(q) = \{s \in (\Sigma \backslash \Sigma_{oj})^*\colon s \in L_\sigma(G, x'_k) \text{ for some } \sigma \in \Sigma_{oi}, \\ i \in \{1, 2\} \backslash \{j\}, \text{ and some } k \in \{1, \ldots, n\}\}.$$

Then the unobservable reach of q with respect to $\Sigma \backslash \Sigma_{oj}$ is defined as follows:

$$UR_j(q) = \{q\} \cup \bigcup_{s \in S_j(q)} \{((y_s, l_s), (y'_s, l'_s))\}$$

where (i) y'_s is the successor of some x'_k, $k \in \{1, \ldots, n\}$, after sub-trace $s \in S_j(q)$, (ii) y_s is the immediate predecessor along s of y'_s in G', and (iii) l_s, l'_s are the failure labels corresponding to y_s, y'_s obtained by propagating the label l'_k of x'_k according to the label propagation function defined in Sampath et al. (1995).

The unobservable reach appends to the components of each state of the extended diagnoser G^e_{dj} some additional components (along with failure labels and predecessors) that may have been reached following an additional event or a sequence of events that are not observable by the local site j. Note here that in the above definition, y_s may not be equal to x'_k. Also note that while we call $UR_j(q)$ the unobservable reach of q with respect to $\Sigma \backslash \Sigma_{oj}$, its definition stipulates that the sub-traces that are used to generate it end with an event in Σ_{oi}, the other set of observable events.

Example 3. Consider the system discussed in Example 1. The extended diagnosers G^e_{d1} and G^e_{d2} associated with the projections P_1 and P_2 are shown in Figure 3. Consider the state $q = \{(1N, 3N), (1N, 4N)\}$ in G^e_{d2}. To compute the unobservable reach of q with respect to $\Sigma \backslash \Sigma_{o2}$, we first find the set $S_2(q) = \{a, c, ac\}$. The successors of state 3 after sub-traces a and ac are 5 and 7, respectively, while the successor of state 4 after sub-trace c is 6. Therefore, $UR_2(q) = \{(1N, 3N), (1N, 4N), (3N, 5N), (5N, 7N), (4N, 6N)\}$. All state labels are N since there were no failure events along any sub-trace. Note here that although state 7 is a successor of state 3 along the sub-trace ac, the immediate predecessor of 7 in G' (not pictured) is state 5, so the corresponding tuple (after adding the failure labels) is $(5N, 7N)$.

To provide the necessary and sufficient conditions of diagnosability in terms of G^e_d, we need the following definitions.

Definition 10. A state $q \in Q^e_d$ is said to be F_i-certain if $\forall (x, l) \in SP(q)$, $F_i \in l$.

Definition 11. A state $q \in Q^e_d$ is said to be F_i-uncertain if $\exists (x, l), (y, l') \in SP(q)$, x not necessarily distinct from y, such that $F_i \in l$ and $F_i \notin l'$.

Definition 12 (Definition 1 in Sampath, 1993). A set of states $q_1, q_2, \ldots, q_n \in Q_d^e$ is said to form a cycle in G_d^e if the following is true:

$$\delta_d^e(q_l, \sigma_l) = q_{(l+1)}, \quad l = 1, 2, \ldots, n-1, \text{ and } \delta_d^e(q_n, \sigma_n) = q_1$$

for some observable events $\sigma_i, i = 1, \ldots, n$.

Definition 13 (Definition 2 in Sampath, 1993). A set of (x_i, l_i) pairs, where $(x_i, l_i) \in Q_o$, $i = 1, 2, \ldots, n$, is said to form a matched cycle in G_d^e if $\exists q_i \in G_d^e$, $i = 1, 2, \ldots, n$, such that:

$$((x_i, l_i), (x_{i+1}, l_{i+1})) \in q_{i+1}, \quad i = 1, 2, \ldots, n-1, \text{ and } ((x_n, l_n), (x_1, l_1)) \in q_1.$$

Note that the existence of such a set of (x_i, l_i) pairs has the two following implications (from the construction procedure of G_d^e):

1. $q_i, i = 1, 2, \ldots, n$, form a cycle in G_d^e.
2. $x_i, i = 1, 2, \ldots, n$, form a cycle in G'.

Definition 14 (Definition 3 in Sampath, 1993). A set of states $q_1, q_2, \ldots, q_n \in Q_d^e$ forming a cycle of F_i-uncertain states in G_d^e is said to form an F_i-indeterminate cycle in G_d^e if the following hold:

1. $\exists$ a set of $(x_j, l_j) \in SP(q_j)$, $j = 1, 2, \ldots, n$, forming a matched cycle in G_d^e, with $F_i \in l_j$, $j = 1, 2, \ldots, n$,

   and

2. $\exists$ a set of $(y_j, l'_j) \in SP(q_j)$, $j = 1, 2, \ldots, n$, forming a matched cycle in G_d^e, with $F_i \notin l'_j$, $j = 1, 2, \ldots, n$.

Next we state a result that relates the existence of F_i-indeterminate cycles in G_d^e to the existence of F_i-indeterminate cycles in G_d.

PROPOSITION 1 *Consider a system G, its diagnoser G_d, and its extended diagnoser G_d^e. Then there are F_i-indeterminate cycles in G_d if and only if there are F_i-indeterminate cycles in G_d^e.*

Proof: Sufficiency($\Leftarrow$). G_d^e has F_i-indeterminate cycles. Consider a set of states q_k, $k = 1, \ldots, n$, that form an F_i-indeterminate cycle in G_d^e. We claim that the set of states $\{p_1, \ldots, p_m\} = SP(\{q_1, \ldots, q_n\})$, $m \leq n$, forms an F_i-indeterminate cycle in G_d. (Note here that $m \leq n$ since, as discussed earlier, one diagnoser state may be associated with more than one extended diagnoser states.) The claim can be established as follows: by assumption, there exist two sets of states of the form $(x_j, l_j), (y_j, l'_j) \in SP(q_j)$, $j = 1, \ldots, m$ such that $F_i \in l_j$, but $F_i \notin l'_j$ (cf. Definition 14). Hence the cycle of states $\{p_1, \ldots, p_m\}$

in G_d is an F_i-uncertain cycle. Moreover, by the implications of Definition 13, the sets $\{x_j\}$ and $\{y_j\}$ form cycles in G'. Therefore the resulting cycle in G_d is F_i-indeterminate by Definition 5.

Necessity($\Rightarrow$) G_d has F_i-indeterminate cycles. From Definition 5, there exist two traces s and s' in $L(G)$, such that $P(s) = P(s')$, $F_i \notin s$, $F_i \in s'$ and s, s' are arbitrarily long. Since $L(G)$ is a regular language, then the fact that s, s' are arbitrarily long implies that the system will loop in a cycle, say A (respectively B) if s (respectively s') is executed. Corresponding to A (respectively B) there exists a cycle of pairs (x_j, l_j) (respectively (y_j, l'_j)) in Q_o, $j = 1, \ldots, n$. Moreover, since $P(s) = P(s')$ (x_j, l_j) and (y_j, l'_j), $j = 1, \ldots, n$, belong to the same set of states $\{q_1, \ldots, q_n\}$ in G_d^e; hence they form matched cycles in G_d^e. By the implications of Definition 13 the states $\{q_1, \ldots, q_n\}$ in G_d^e form a cycle, and the fact that $F_i \notin l_j$ but $F_i \in l'_j$ implies that the cycle is F_i-indeterminate. ■

Based on Proposition 1 and Definition 1 we provide a test to check the diagnosability of a language in terms of the extended diagnoser G_d^e:

THEOREM 2[1] *A prefix-closed and live language L is diagnosable with respect to the projection P and the failure partition Π_f on Σ_f if and only if its extended diagnoser G_d^e satisfies the following condition: there are no F_i-indeterminate cycles in G_d^e for all failure types F_i.*

Proof: The proof is a direct consequence of Definition 1 and Proposition 1. ■

Having presented the type of diagnostic information generated at the local sites, along with some of its properties, we next define the communication rules used by the diagnosers.

4.1.2. Communication Rules

To define the communication rules, we first note that right after the occurrence of an event that is observable only by one site, say i, the state of the extended diagnoser at site $j \neq i$ does not contain the true system state. Therefore, for the purpose of communicating information from a local site to the coordinator, we need to augment the state of the extended diagnoser with some additional information, the unobservable reach. We define the communication rules **CR** := (**CR1**, **CR2**) as follows:

- **[CRi]**, $i = 1, 2$: After the agent at site i observes an event $\sigma \in \Sigma_{oi}$, it communicates to the coordinator the corresponding state q_i of its extended diagnoser G_{di}^e, its unobservable reach $UR_i(q_i)$ with respect to $\Sigma \setminus \Sigma_{oi}$, and a status bit, $\mathbf{SB_i}$, that takes the values $\mathbf{SB_i} = 1$ when $\sigma \in \Sigma_{oj}$, $j \in \{1, 2\}$, $j \neq i$, or $\mathbf{SB_i} = 0$ when $\sigma \notin \Sigma_{oj}$.

4.1.3. Decision Rule

The decision rule of the coordinator consists of two components: (1) a rule according to which its information is updated; and (2) a rule according to which failure occurrences are declared and broadcast to the failure recovery module.

As stated earlier, the coordinator declares that a failure of type F_i has occurred when its diagnostic information C is F_i-certain (cf. Definition 6). To specify the information update rule we first need to define the following operators (Definitions 15 and 16).

Definition 15. Let $q_1 = \{((x_1, l_1), (x'_1, l'_1)), \ldots, ((x_n, l_n), (x'_n, l'_n))\}$ and $q_2 = \{((y_1, l_1), (y'_1, l'_1)), \ldots, ((y_m, l_m), (y'_m, l'_m))\}$ belong to $Q_o \times Q_o$. We denote by $\cap_e^i$, $i \in \{L, R\}$ the intersection scheme that acts on q_1 and q_2, and we define it as follows:

$$q_1 \cap_e^i q_2 \stackrel{\triangle}{=} \{((z, l), (z', l')) \in Q_o \times Q_o\colon\ (z', l') = (x'_i, l'_i) = (y'_j, l'_j) \text{ for some } i,\ j,\ i \in \{1, 2, \ldots, n\},\ j \in \{1, 2, \ldots, m\},\ and (z, l) = (x_i, l_i) \text{ if } i = L,\ \text{otherwise } (z, l) = (y_j, l_j)\}.$$

This intersection scheme is a regular intersection of the components of the two system state estimates along with their failure labels. However, the intersection applies to the components corresponding to the current system state estimates and not to their immediate predecessors. The components of $q_1 \cap_e^i q_2$ corresponding to the immediate predecessors are determined by operator i. The intersection scheme $\cap_e^i$ introduced by Definition 15 is illustrated by the following example:

Example 4. Let $q_1 = \{(6N, 8N), (7N, 10F1)\}$ and $q_2 = \{(3N, 11N), (3N, 10F1), (4N, 8N)\}$. To compute $q_1 \cap_e^L q_2$ we find the common components in the two current system state estimates, namely $8N$ and $10F1$, and we append the predecessors of $8N$ and $10F1$ in q_1 to the states to get $q_1 \cap_e^L q_2 = \{(6N, 8N), (7N, 10F1)\}$. Similarly, $q_1 \cap_e^R q_2 = \{(4N, 8N), (3N, 10F1)\}$.

The second operator we introduce is another intersection scheme, and is defined as follows:

Definition 16. Let $q_1 = \{((x_1, l_1), (x'_1, l'_1)), \ldots, ((x_n, l_n), (x'_n, l'_n))\}$ and $q_0 = \{((y_1, l_1), (y'_1, l'_1)), \ldots, ((y_m, l_m), (y'_m, l'_m))\}$ belong to $Q_o \times Q_o$. We denote by $\cap_c$ the intersection scheme that acts on q_1 and q_0, and we define it as follows:

$$q_1 \cap_c q_0 \stackrel{\triangle}{=} \{((z, l), (z', l')) \in Q_o \times Q_o\colon\ (z, l) = (x_i, l_i) = (y'_j, l'_j),\ \text{for some } i,\ j,\ i \in \{1, 2, \ldots, n\},\ j \in \{1, 2, \ldots, m\},\ \text{and } (z', l') = (x'_i, l'_i)\}.$$

The intersection scheme $\cap_c$ is illustrated by the following example:

Example 5. Let $q_1 = \{(6N, 8N), (7N, 10F1)\}$ and $q_0 = \{(4N, 6N)\}$. $q_1 \cap_c q_0 = \{(6N, 8N)\}$ since the component $10F1$ of q_1 was reached from the component $7N$ which is not present in q_0.

In addition to the above operators, we need to describe the structure of the coordinator before we precisely specify its information update rule. In addition to the register C where the coordinator stores its current diagnostic information, eight supplementary registers are

Table 1. Information update rule at the coordinator site (Protocol 1).

	Last report received from G^e_{d1}					
	SB	SB_1	C	New SB	New SB_{1old}	New SB_{2old}
DR1	0	0	$(R_1 \cap^i_e R_4) \cap_c C_{old}$	0	1	0
DR2	0	1	Wait	1	Unmodified	Unmodified
—	1	0	Impossible	—	—	—
DR3	1	1	$(R_1 \cap^i_e R_2) \cap_c C_{old}$	0	1	1
	Last report received from G^e_{d2}					
	SB	SB_2	C	New SB	New SB_{1old}	New SB_{2old}
DR4	0	0	$(R_2 \cap^i_e R_3) \cap_c C_{old}$	0	0	1
DR5	0	1	Wait	1	Unmodified	Unmodified
—	1	0	Impossible	—	—	—
DR6	1	1	$(R_1 \cap^i_e R_2) \cap_c C_{old}$	0	1	1

the i superscript in $\cap^i_e$ depends on the current values of the flip-flops SB_{1old} and SB_{2old}, not shown in this table

used for storing messages and previous relevant values necessary for the update of its information. These registers are: $R1$, $R2$, $R3$, $R4$, C_{old}, SB, SB_{1old}, and SB_{2old}. $R1$ and $R2$ hold the latest states of G^e_{d1} and G^e_{d2}, respectively, $R3$ and $R4$ hold the latest unobservable reaches of G^e_{d1} and G^e_{d2}, respectively, C_{old} holds the previous coordinator diagnostic information, SB specifies whether the last observed event is observed by both sites (1) or not (0) and SB_{1old}, SB_{2old} provide necessary information to compute the new coordinator diagnostic information.

The information update rule is given in Table 1. The rule picks one of the actions **DR1**–**DR6** depending on the available information, i.e., which site observed the last and previous to the last events, and who sent the last message to the coordinator.

The rationale behind the actions **DR1** to **DR6** can be summarized as follows. Once a message from one diagnoser, say G^e_{d1}, reaches the coordinator after the occurrence of an observable event, the state of that diagnoser should contain the true system state. Moreover, if the message says that the event is not observed by the other site (site 2), the current unobservable reach of the diagnoser G^e_{d2} also contains the true system state. Consequently, the logical action is to intersect the state of G^e_{d1} with the unobservable reach of G^e_{d2} using the intersection scheme $\cap^i_e$ (the bits SB_{1old} and SB_{2old} specify the value of i in $\cap^i_e$: if $SB_{1old} = 1$, then $i = L$, that is you append the predecessors from the state of G^e_{d1}; otherwise $i = R$), and then intersect the result with the old coordinator diagnostic information, using the intersection scheme $\cap_c$, to generate the new coordinator diagnostic information. The last intersection is needed to eliminate the possibility of including any illegal behavior in the coordinator diagnostic information. In case the event is also observed by site 2, the state of G^e_{d2} contains the true system state. Therefore, the logical action in this case is to intersect the states of the diagnosers G^e_{d1} and G^e_{d2} by applying the $\cap^i_e$ intersection, and then refine the result by applying the intersection scheme $\cap_c$ as discussed earlier. Note here that before performing any update of the coordinator diagnostic information, the current coordinator

diagnostic information is saved into the register C_{old} for later use. Also, the flip-flops are modified once the update of the coordinator diagnostic information is completed. At reset, *R1* and *R2* are initialized with the initial states of G^e_{d1} and G^e_{d2}, respectively, and *R3* and *R4* hold the initial unobservable reaches of G^e_{d1} and G^e_{d2}, respectively.

Note that the coordinator is not aware of the rationale described above when it updates its diagnostic information and when it declares that a failure of a certain type has occurred. The coordinator simply executes the operations $\cap^i_e$, $\cap_c$, updates all of its registers, and declares the occurrence of failures according to the decision rule described above.

In summary, the registers of the coordinator are updated according to the information update rule presented in Table 1. Once C, the coordinator's diagnostic information is F_i-certain, the coordinator broadcasts to the failure recovery module that a failure of type F_i has occurred.

4.2. Diagnostic Properties of Protocol 1

The diagnostic properties of Protocol 1 are summarized by Theorem 3, the proof of which is based on the following proposition.

PROPOSITION 2 *Let q_1, q_2, and q be the states of the extended diagnosers G^e_{d1}, G^e_{d2}, and G^e_d, respectively, after the system executed the trace $s = s_1 a u b$, where $a, b \in \Sigma_o$ $(= \Sigma_{o1} \cup \Sigma_{o2})$, $u \in \Sigma^*_{uo}$. Denote by q_{old} the state of G^e_d after the execution of $s_1 a$. Then the following is true:*

1. *if $b \in \Sigma_{o1} \cap \Sigma_{o2}$ then*

$$(i) \quad q = (q_1 \cap^L_e q_2) \cap_c q_{old}, \text{ if } a \in \Sigma_{o1}$$
$$(ii) \quad q = (q_1 \cap^R_e q_2) \cap_c q_{old}, \text{ otherwise}$$

2. *if $b \in \Sigma_{o1} \backslash \Sigma_{o2}$ then*

$$(i) \quad q = (q_1 \cap^L_e U R_2(q_2)) \cap_c q_{old}, \text{ if } a \in \Sigma_{o1}$$
$$(ii) \quad q = (q_1 \cap^R_e U R_2(q_2)) \cap_c q_{old}, \text{ otherwise}$$

3. *if $b \in \Sigma_{o2} \backslash \Sigma_{o1}$ then*

$$(i) \quad q = (U R_1(q_1) \cap^L_e q_2) \cap_c q_{old}, \text{ if } a \in \Sigma_{o1}$$
$$(ii) \quad q = (U R_1(q_1) \cap^R_e q_2) \cap_c q_{old}, \text{ otherwise.}$$

Proof:

Proof of 1. We first note that

$$SP(q) \subseteq SP(q_i) \subseteq SP(U R_i(q_i)), i = \{1, 2\}. \tag{18}$$

The first inclusion is true since the set of observable events Σ_{oi} is a subset of the original set of observable events Σ_o and $b \in \Sigma_{o1} \cap \Sigma_{o2}$, and the second inclusion is true by definition (cf. Definition 9). In case (i) we have

$$(q_1 \cap_e^L q_2) \cap_c q_{old} = (q_1 \cap_c q_{old}) \cap_e^L q_2 \tag{19}$$

$$(q_1 \cap_c q_{old}) = q. \tag{20}$$

(19) follows from the definition of the intersection operators $\cap_c$ and $\cap_e^L$. (20) is obtained as follows: by definition, $q_1 \cap_c q_{old}$ gives all state estimate tuples in q_1 that are reached by the observable event b. Since b is the next observable event after a in s and $SP(q) \subseteq SP(q_1)$ by (18), $q_1 \cap_c q_{old}$ is the state of the diagnoser G_d^e which is q by definition. Combining (19) and $SP(q) \subseteq SP(q_2)$ from (18) we obtain $q = (q_1 \cap_e^L q_2) \cap_c q_{old}$. To prove case (ii), we note that by Definition 15 we can write $q_1 \cap_e^L q_2 = q_2 \cap_e^R q_1$. So by exchanging the roles of q_1 and q_2, and using the same arguments as in case (i) we have $q = (q_1 \cap_e^R q_2) \cap_c q_{old}$.

Proof of 2. We note first that

$$SP(q) \subseteq SP(q_1) \text{ and } SP(q) \subseteq SP(UR_2(q_2)). \tag{21}$$

The inclusions are true since the set of observable events Σ_{oi} is a subset of the original set of observable events Σ_o and $b \in \Sigma_{o1} \backslash \Sigma_{o2}$. The proof of case (i) proceeds in the same way as the proof of $1-(i)$ with the minor modification of using $UR_2(q_2)$ instead of q_2. To prove (ii) we have

$$(q_1 \cap_e^R UR_2(q_2)) \cap_c q_{old} = q_1 \cap_e^R (UR_2(q_2) \cap_c q_{old}) \tag{22}$$

$$(UR_2(q_2) \cap_c q_{old}) = q. \tag{23}$$

(22) follows from the definition of the intersection operators $\cap_c$ and $\cap_e^R$. (23) is obtained as follows: by definition, $UR_2(q_2) \cap_c q_{old}$ gives all state estimate tuples in $UR_2(q_2)$ that are reached by the observable event b. This is true by Definition 9: $UR_2(q_2)$ may include state estimate tuples $\{(x, l), (x', l')\}$ whose current state x' may be reached by a sequence of observable events and not only by one observable event, like in the case of the event b; however in such a case the predecessor state x is by definition the immediate predecessor of x' in G', and this predecessor does not belong to any $SP(x_i)$, where $x_i \in q_{old}$. Since b is the next observable event after a in s and $SP(q) \subseteq SP(UR_2(q_2))$ by (21), $UR_2(q_2) \cap_c q_{old}$ is the state of the diagnoser G_d^e which is q by definition. Combining (22) and $SP(q) \subseteq SP(q_1)$ from (21) we obtain $q = (q_1 \cap_e^R UR_2(q_2)) \cap_c q_{old}$.

Proof of 3. Exchange the roles of q_1 and $UR_1(q_1)$ with q_2 and $UR_2(q_2)$, respectively, and proceed as in the proof of 2. ■

Proposition 2 can be used to prove the main result concerning the diagnostic properties of Protocol 1.

THEOREM 3 *(i) The coordinator's diagnostic information C under Protocol 1 is the same as the state of the centralized extended diagnoser* G_d^e.

(ii) Protocol 1 achieves the same diagnostic performance as a centralized diagnoser.

Proof: (i) We prove part (i) by induction on the number of observable events (in $\Sigma_o = \Sigma_{o1} \cup \Sigma_{o2}$) in the trace s.

Basis of induction: Let $|P(s)| = |b| = 1$. In this case $C_{old} = \{(x_0, N), (x_0, N)\}$ by assumption, where x_o is the initial state of the system. Moreover, by assumption both G_{d1}^e and G_{d2}^e have the same initial state $\{(x_0, N), (x_0, N)\}$. If $b \in \Sigma_{o1} \cap \Sigma_{o2}$ then $q_1 = q_2 = q$ by the construction of the diagnosers. Therefore $q_1 \cap_e^i q_2 = q$ and $q \cap_c C_{old} = q$ by definition. If $b \in \Sigma_{o1} \backslash \Sigma_{o2}$ then $q_1 = q$ by construction and $q \subseteq UR_2(q_2)$ as discussed earlier in the proof of Proposition 2. Therefore, $q_1 \cap_e^i UR_2(q_2) = q$, and $q \cap_c C_{old} = q$ by definition. The proof of the case when $b \in \Sigma_{o2} \backslash \Sigma_{o1}$ is symmetric to the case where $b \in \Sigma_{o1} \backslash \Sigma_{o2}$.

Induction step: The proof of the induction step is provided by Proposition 2 since by Assumptions **A4** and **A5** every message is received in the order it was sent.

(ii) From part (i) and the specification of the coordinator's decision rule it follows that Protocol 1 achieves the same diagnostic performance as a centralized diagnoser. ■

Note that, according to Assumption **A8**, the coordinator has no knowledge of the system model, and has limited memory and limited processing capabilities. Yet, if the coordinator has the memory and processing capabilities required by the decision rule described in Section 4.1.3, it can diagnose the same types of failures as a centralized diagnoser; by receiving the extended diagnoser states (and unobservable reaches) and using the rules $\cap_e^i$, $i = L, R$ and $\cap_c$ the coordinator, in essence, can keep track of the state of the system in the same way as the centralized diagnoser. Consequently, it has the same diagnostic properties as the centralized diagnoser.

4.3. Necessary and Sufficient Conditions for Diagnosability

In Section 4.2, we showed that the information update rule that is used at the coordinator site is reconstructing the centralized diagnoser state. Consequently the necessary and sufficient conditions for diagnosability with respect to Protocol 1 can be stated with respect to the centralized diagnoser as follows:

THEOREM 4 *A live and prefix-closed language L is diagnosable with respect to Protocol 1, the set of projections* P_1, P_2 *and the failure partition* Π_f *on* Σ_f *if and only if the diagnoser* G_d *does not have* F_i*-indeterminate cycles for all failure types* F_i.

Proof: Sufficiency($\Leftarrow$). Suppose G_d does not have F_i-indeterminate cycles. Then by Proposition 1, G_d^e does not have F_i-indeterminate cycles. Consider a trace $st \in L(G)$ such that $s \in \Psi(\Sigma_{fi})$, and t is long enough, i.e., $\|t\| > n$, where n can be arbitrarily large. Then, by assumption, st', $t' \in \overline{t}$ does not lead to an F_i-indeterminate cycle in G_d^e. Consequently, an argument similar to the one used in the proof of Theorem 2 in Sampath et al. (1995)

Table 2. Illustration of the application of Protocol 1.

Event	$R1$	$R3$	$R2$	$R4$	C_{old}	C
ϵ	(1N,1N)	(1N,1N), (1N,3N)	(1N,1N)	(1N,1N), (1N,2N)	(1N,1N)	(1N,1N)
a	(1N,2N), (1N,5N)	(1N,2N), (1N,5N), (2N,4N)	(1N,1N)	(1N,1N), (1N,2N)	(1N,1N)	(1N,2N)
b	(1N,2N), (1N,5N)	(1N,2N), (1N,5N), (2N,4N)	(1N,3N), (1N,4N)	(1N,3N), (1N,4N), (3N,5N), (5N,7N), (4N,6N)	(1N,2N)	(2N,4N)
c	(2N,6N), (5N,7N)	(2N,6N), (5N,7N)	(1N,3N), (1N,4N)	(1N,3N), (1N,4N), (3N,5N), (5N,7N), (4N,6N)	(2N,4N)	(4N,6N)
d	(6N,8N), (7N,10F1)	(6N,8N), (7N,10F1)	(3N,11N), (3N,10F1), (4N,8N)	(3N,11N), (3N,10F1), (4N,8N)	(4N,46)	(6N,8N)

shows that G_d^e will enter an F_i-certain state within a finite number of steps, say n_i'. This implies by Theorem 3 that the coordinator's diagnostic information C will be F_i-certain within a finite number of steps equal to n_i'. Therefore, $L(G)$ is diagnosable with respect to Protocol 1.

Necessity($\Rightarrow$). We prove the contrapositive. Assume that G_d does enter a F_i-indeterminate cycle. Then by Proposition 1 G_d^e enters an F_i-indeterminate cycle. This implies that the coordinator's diagnostic information C, which is equal to the centralized diagnoser state, will remain F_i-uncertain for an arbitrarily long number of steps. Hence, L is not diagnosable with respect to Protocol 1. ■

We conclude this section with an example that illustrates the application of Theorem 4.

Example 6. Consider again the system shown in Figure 2 with $\Sigma = \{a, b, c, d, e, \sigma\}$, $\Sigma_{uo} = \{\sigma\}$, $\Sigma_{f1} = \{\sigma\}$, $\Sigma_{o1} = \{a, c, d, e\}$, and $\Sigma_{o2} = \{b, d, e\}$. The diagnoser G_d for this system is shown in Figure 4. The only cycle whose states carry failure labels is $\{10F1, 9F1\}$; however its states are F_1-certain; hence there are no F_1-indeterminate cycles, and by Theorem 4 the system is diagnosable by Protocol 1.

Table 2 illustrates the application of Protocol 1 when the system executes the trace $abcd$. For instance, after the message from site 2 regarding the occurrence of event b reaches the coordinator and all operations are performed we have $SB = 0$, $SB_{1old} = 0$ and $SB_{2old} = 1$ (the bits SB, SB_{1old}, and SB_{2old} are not shown in the table for space limitation). Once the message regarding the occurrence of the event c reaches the co-

ordinator, the following occurs: since the event is only observed by site 1 and since $SB = 0$, action **DR1** is taken; therefore, the current $C = (2N, 4N)$ is saved into C_{old}, and since $SB_{1old} = 0$ then $C = (R_1 \cap_e^R R_4) \cap_c C_{old} = (\{(2N, 6N), (5N, 7N)\} \cap_e^R \{(1N, 3N), (1N, 4N), (3N, 5N), (5N, 7N), (4N, 6N)\}) \cap_c (2N, 4N) = \{(4N, 6N), (5N, 7N)\} \cap_c (2N, 4N) = (4N, 6N)$. The registers SB, SB_{1old} and SB_{2old} are set to 0, 1, 0, respectively. The next observable event is d and it is observed by both sites; assume that the report from site 2 about the occurrence of event d, reaches the coordinator before the one form site 1. Once the report from site 2 reaches the coordinator, action **DR5** is executed, i.e., SB is set to 1, because $SB = 0$. When the report from site 1 reaches the coordinator, action **DR3** is executed, which means that $C = (R_1 \cap_e^L R_2) \cap_c C_{old} = (\{(6N, 8N), (7N, 10F1)\} \cap_e^L \{(3N, 11N), (3N, 10F1), (4N, 8N)\}) \cap_c (4N, 6N) = \{(6N, 8N), (7N, 10F1)\} \cap_c (4N, 6N) = (6N, 8N)$. The registers SB, SB_{1old} and SB_{2old} are set to 0, 1, 1, respectively.

We emphasize the need to incorporate $\cap_c$ and C_{old} in the decision rule to guarantee the performance of Protocol 1. For that matter, consider that the system has just executed the trace $abcd$. The last row of Table 2 describes the content of all the registers at the coordinator site after all messages related to the occurrence of event d have reached the coordinator site. If we compute C by only using $\cap_e^i$, $i = L, R$, i.e., $C = R_1 \cap_e^L R_2$ we get $C = \{(6N, 8N), (7N, 10F1)\}$, which clearly is not equal to the state of the centralized diagnoser $(6N, 8N)$. However, if we incorporate $\cap_c$ in the decision rule we indeed reconstruct the state of the centralized diagnoser. The operator $\cap_c$ eliminates in an extended diagnoser state all current state estimates whose predecessor state estimates are not current state estimates in the old coordinator state. Since $\cap_c$ is used after the occurrence of every observable event, then by an inductive argument one can visualize that we are "memorizing" the past. In other words, it is not sufficient to retain a one-step memory as provided by the extended diagnoser; rather, a "n-step" memory, where n represents the length of the observed trace, is needed, and that is achieved by incorporating $\cap_c$ in the decision rule.

4.4. Discussion

We first note that the partitioning of observable events does not affect the diagnostic capabilities of Protocol 1: irrespective of the partitioning of the set of observable events Σ_o, as long as the centralized diagnoser is capable of identifying all failure types, so is Protocol 1, and vice-versa. This is a direct consequence of Theorem 3.

Having demonstrated that Protocol 1 is capable of diagnosing all failure types that are diagnosed by a centralized diagnoser, one may ask the following question: Is it possible to replace the extended diagnosers G_{d1}^e and G_{d2}^e by the diagnosers G_{d1} and G_{d2}, respectively, while maintaining the same fundamental structure, i.e., the same functional form of the communication rules and the coordinator's decision rule (with the obvious modifications dictated by the change from extended diagnosers to diagnosers) so that the resulting protocol achieves the same diagnostic performance as Protocol 1? The following example shows that a modification such as the above does not lead to a protocol with the same diagnostic capabilities as Protocol 1.

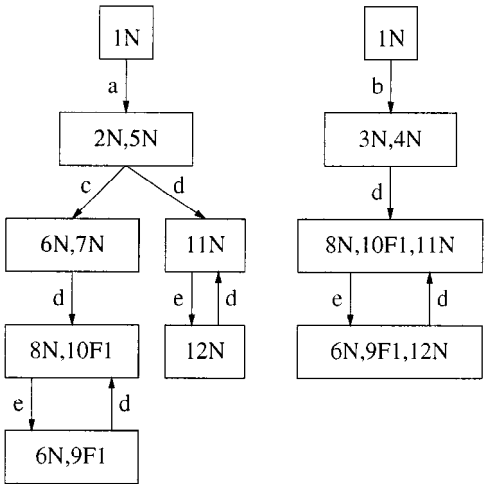

Figure 5. The local diagnosers for Example 1.

Example 7. Consider the system discussed in Example 1 and shown in Figure 2 with $\Sigma = \{a, b, c, d, e, \sigma\}$, $\Sigma_{uo} = \{\sigma\}$, $\Sigma_{f1} = \{\sigma\}$, $\Sigma_{o1} = \{a, c, d, e\}$, and $\Sigma_{o2} = \{b, d, e\}$. We showed in Example 6 that this system is diagnosable with respect to Protocol 1. Consider now the following protocol, say Protocol 2: the diagnosers G_{d1}, G_{d2} (cf. Figure 5) replace G^e_{d1}, G^e_{d2}, respectively. The fundamental structure, i.e., the functional form of the communication rules and the coordinator's decision rule remain the same as in Protocol 1. The modifications resulting from replacing G^e_{d1}, G^e_{d2} with G_{d1}, G_{d2}, respectively, lead to the following rules: the diagnosers at the local sites communicate their states, their unobservable reaches and a status bit (as in Protocol 1) to the coordinator. The coordinator declares that a failure of type F_i has occurred when its diagnostic information C is F_i-certain (cf. Definition 6); it broadcasts this decision to the failure recovery module. The coordinator's information update rule results from the information update rule of Protocol 1 by replacing $\cap^i_e$ with the regular set intersection and eliminating $\cap_c$. We will give a detailed description of this protocol in the next section. Assume now that the system is executing the trace $abcdede\ldots$; then G_{d1} and G_{d2} are looping simultaneously in the cycles $\{(8N, 10F1), (6N, 9F1)\}$ and $\{(8N, 10F1, 11N), (6N, 9F1, 12N)\}$, respectively. Moreover these cycles are F_1-indeterminate in their respective diagnosers: traces[2] $s = abc(de)^*$ and $s' = bac\sigma(de)^*$ both lead to the two cycles (since $P_1(s) = P_1(s') = ac(de)^*$ and $P_2(s) = P_2(s') = b(de)^*$) and σ, the failure of type F_1, belongs to s' while it does not belong to s. Under Protocol 2, the coordinator is not able to differentiate between the two traces: the state of the coordinator is either (8N,10F1), after the occurrence of the event d, or $(6N, 9F1)$, after the occurrence of the event e, and this continues indefinitely. We refer to such problem as the "ordering problem" in untimed DES, since using the available diagnostic information, the coordinator is not capable of "ordering" the events in traces s and s', i.e., whether the event that occurred first is a or b. In Protocol 1, this problem was

solved by the use of the diagnostic information generated by extended diagnosers at the local sites and the decision rule of the coordinator presented in Section 4.1.3.

Since the above example shows that the objective of diagnosing all failure types that are diagnosed using the centralized diagnoser cannot be achieved using Protocol 2, one may seek conditions on the system model G under which this objective can be met. This is discussed in the next section.

5. A Second Coordinated Decentralized Protocol: Protocol 2

5.1. Objective and Assumptions

In this section we present a protocol, called Protocol 2, that has the following features: (i) it employs diagnosers at the local sites (instead of extended diagnosers); (ii) it maintains the same functional form of the communication and decision rules as in Protocol 1; (iii) under certain conditions, identified below, it achieves the same performance as the centralized diagnoser.

To identify conditions under which Protocol 2 achieves the same performance as the centralized diagnoser, we introduce the notions of *state-ambiguous* and *failure-ambiguous* traces with respect to the projections P_1 and P_2. State-ambiguous traces are defined as follows.

Definition 17. A trace $s \in L(G)$ is said to be state-ambiguous with respect to the projections P_1 and P_2 if there exist two traces, s' and s'' in $L(G)$ such that s' and s'' are arbitrarily long, not necessarily distinct, and the following is true:

1. $P_1(s) = P_1(s')$ but $P(s) \neq P(s')$,
2. $P_2(s) = P_2(s'')$ but $P(s) \neq P(s'')$,
3. s' and s'' share the same failure properties, i.e., a failure of type F_i, $i \in \{1, \ldots, m\}$, belongs to s' if and only if a failure of type F_i (not necessarily the same failure event) belongs to s''.

This definition says that the traces s, s' and s, s'' can be distinguished under the projection P; however s and s' are not distinguishable under P_1 while s and s'' are not distinguishable under P_2. Furthermore, s' and s'' have similar failure properties. Thereafter when we refer to a trace as being state-ambiguous, the projections P_1 and P_2 will be understood from the context. The concept of "state-ambiguous traces" is illustrated by the following example.

Example 8. Consider the system of Example 1 shown in Figure 2. The set of events is $\Sigma = \{a, b, c, d, e, \sigma\}$, and σ is the only unobservable and failure event. $\Sigma_{o1} = \{a, c, d, e\}$ and $\Sigma_{o2} = \{b, d, e\}$. If we consider the traces $s = bac\sigma(de)^*$ and $s' = s'' = abc(de)^*$, then s is state-ambiguous since (1) $P_1(s) = P_1(s') = ac(de)^*$ but $P(s) = bac(de)^* \neq$

$abc(de)^* = P(s')$, (2) $P_2(s) = P_2(s'') = b(de)^*$ but $P(s) = bac(de)^* \neq abc(de)^* = P(s'')$ and (3) s', s'' being equal share the same failure properties. Example 11, presented in Section 5.3, provides another example of a state-ambiguous trace where $s' \neq s''$.

Failure-ambiguous traces are defined as follows.

Definition 18. A trace s $\in L(G)$ is said to be failure-ambiguous with respect to the projections P_1 and P_2 and the failure type F_i if there exist two traces, s' and s'' in $L(G)$ such that s' and s'' are arbitrarily long, not necessarily distinct, and the following is true:

1. $P_1(s) = P_1(s')$ but $P(s) \neq P(s')$,
2. $P_2(s) = P_2(s'')$ but $P(s) \neq P(s'')$,

3a. $F_i \in s$ but $F_i \notin s'$.

3b. $F_i \in s$ but $F_i \notin s''$.

4. s' and s'' share the same failure properties, i.e., a failure of type F_j, $j \in \{1, \ldots, m\}$, $j \neq i$, belongs to s' if and only if a failure of type F_j (not necessarily the same failure event) belongs to s''.

A failure-ambiguous trace s is also state-ambiguous. However, not every state-ambiguous trace is failure-ambiguous since for that to be true s' and s'' should not share the same failure type F_i with s. Therefore, by definition, we have that the class of failure-ambiguous traces is a subset of the class of state-ambiguous traces. Thereafter when we refer to a trace as being failure-ambiguous, the projections P_1 and P_2 and the failure type F_i will be understood from the context. The concept of "failure-ambiguous traces" is illustrated by the following example.

Example 9. Consider the system of Example 1 shown in Figure 2. The set of events is $\Sigma = \{a, b, c, d, e, \sigma\}$, and σ is the only unobservable and failure event. $\Sigma_{o1} = \{a, c, d, e\}$ and $\Sigma_{o2} = \{b, d, e\}$. If we consider the traces $s = bac\sigma(de)^*$ and $s' = s'' = abc(de)^*$, then s is failure-ambiguous since we showed in Example 8 that s is state-ambiguous and moreover $s' = s''$ exhibit only normal behavior while s has a failure of type F_1 (conditions 3*a*, 3*b*). Example 11, presented in Section 5.3, provides another example of a failure-ambiguous trace where $s' \neq s''$.

We will study the performance of Protocol 2 under Assumptions **A1**–**A8** (cf. Section 3.1) and one of the following additional assumptions.

A9 There are no state-ambiguous traces in $L(G)$.

A9′ There are no failure-ambiguous traces (with respect to all failure types) in $L(G)$.

The study of the performance of Protocol 2 under the set of assumptions **A1**–**A9** is performed for the purpose of comparing its performance to that of Protocol 1. The study of

the performance of Protocol 2 under the set of assumptions **A1**–**A8** and **A9**′ is performed for the purpose of comparing its performance to that of Protocol 3, introduced later in Section 6.

5.2. Specification of the Protocol

In this section, we present in detail Protocol 2, a protocol that realizes the coordinated decentralized architecture presented in Section 3.1. The specification of the protocol is done under the Assumptions **A1**–**A8** of Section 3.1.

5.2.1. Diagnostic Information at Local Sites

We begin by specifying the type of diagnostic information generated at local sites. Since diagnosers are implemented at local sites, then the diagnostic information available at each site is provided by the state of the diagnoser. The state information is refined by the *unobservable reach* which is defined as follows.

Definition 19. Let $q = \{(x_1, l_1), \ldots, (x_n, l_n)\}$ be a diagnoser state. Define the set

$$S_j(q) = \{s \in (\Sigma \backslash \Sigma_{oj})^* \colon s \in L_\sigma(G, x_k) \text{ for some } \sigma \in \Sigma_{oi}, i \in \{1, 2\} \backslash \{j\}, \text{ and some } k \in \{1, \ldots, n\}\}.$$

Then the unobservable reach of q with respect to $\Sigma \backslash \Sigma_{oj}$ is defined as follows:

$$UR_j(q) = \{q\} \cup \bigcup_{s \in S_j(q)} \{(y_s, l_s)\}$$

where (i) y_s is the successor of some x_k, $k \in \{1, \ldots, n\}$, after sub-trace $s \in S_j(q)$, and (ii) l_s is the failure label corresponding to y_s, obtained by propagating the label l_k of x_k according to the label propagation function defined in Sampath et al. (1995).

The unobservable reach of a diagnoser at a state represents all possible states where the system may be after the execution of a trace in the language. Note that by definition the diagnoser state only represents those states that are reached following an observable event; the unobservable reach appends to the diagnoser state the states that are reached through unobservable events following that observable event up to an event observable by the other site. The following example illustrates the concept of unobservable reach of a diagnoser state.

Example 10. Consider the system discussed in Examples 1 and 8. The diagnosers G_{d1} and G_{d2} associated with the projections P_1 and P_2 are shown in Figure 5. We assume that the state of diagnoser G_{d2} is $q = \{3N, 4N\}$ and compute the unobservable reach of q with respect to $\Sigma \backslash \Sigma_{o2}$. We first find the set $S = \{a, c, ac\}$. The states that are reached from states 3 and 4 are: state 5 through event a, state 6 through event c, and state 7 through the sequence of events ac. Therefore, $UR_2(q) = \{3N, 4N, 5N, 6N, 7N\}$. All states carry the normal label N since there were no failure events along any sub-trace in S.

Table 3. Information update rule at the coordinator site (Protocol 2).

Last report received from G_{d1}					**Last report received from G_{d2}**				
Rule	*SB*	SB_1	*C*	*New SB*	**Rule**	*SB*	SB_2	*C*	*New SB*
DR1	0	0	$R_1 \cap R_4$	0	**DR4**	0	0	$R_2 \cap R_3$	0
DR2	0	1	Wait	1	**DR5**	0	1	Wait	1
–	1	0	Impossible	-	–	1	0	Impossible	-
DR3	1	1	$R_1 \cap R_2$	0	**DR6**	1	1	$R_1 \cap R_2$	0

5.2.2. *Communication Rules*

To define the communication rules, we first note that right after the occurrence of an event that is observable only by one site, say i, the state of the diagnoser at site $j \neq i$ does not contain the true system state. Therefore, to efficiently communicate information to the coordinator, each local site must augment the state of its diagnoser with some additional information, the unobservable reach. We define the communication rules $\mathbf{CR} := (\mathbf{CR1}, \mathbf{CR2})$ as follows:

- **[CRi]**, $i = 1, 2$: After the agent at site i observes an event $\sigma \in \Sigma_{oi}$, it communicates to the coordinator the corresponding state q_i of its diagnoser G_{di}, its unobservable reach $UR_i(q_i)$ with respect to $\Sigma \backslash \Sigma_{oi}$, and a status bit, $\mathbf{SB_i}$, that takes the values $\mathbf{SB_i} = 1$ when $\sigma \in \Sigma_{oj}$, $j \in \{1, 2\}$, $j \neq i$, or $\mathbf{SB_i} = 0$ when $\sigma \notin \Sigma_{oj}$.

5.2.3. *Decision Rule*

The decision rule of the coordinator consists of two components : (1) a rule according to which its information is updated; and (2) a rule according to which failure occurrences are declared and broadcast to the failure recovery module.

To specify the information update rule we first describe the structure of the coordinator. The coordinator has five registers, (*R1, R2, R3, R4, SB*), besides the register C that holds its diagnostic information. The five registers are used to store incoming messages from the local sites and previous relevant values necessary for the update of its information. R_1 and R_2 hold the latest states of G_{d1} and G_{d2}, respectively, R_3 and R_4 hold the latest unobservable reaches of G_{d1} and G_{d2}, respectively, and SB specifies whether to apply the information update rule to the available information in the registers ($SB = 0$) or wait for the next incoming message ($SB = 1$). At reset, *R1* and *R2* are initialized with the initial states of G_{d1} and G_{d2}, respectively, while *R3* and *R4* hold the unobservable reaches of the initial state of G_{d1} and G_{d2}, respectively. The register SB is initially set to 0. The information update rule is specified in Table 3. Based on the available information, the rule picks one of the actions **DR1**–**DR6**. The rationale behind the actions is the following: to compute C we intersect the states of the diagnosers if both sites have observed the last event. In case the true system state is not in the diagnoser state (since the last observable event was not seen by the diagnoser), we use the unobservable reach of the diagnoser instead of its state

in the intersection with the other diagnoser's state. We finally note that the coordinator is not aware of the rationale behind its actions; it simply updates its information and declares the occurrence of failures according to the decision rule described above.

The coordinator declares that a failure has occurred when its diagnostic information C is F_i-certain (cf. Definition 6).

5.3. Diagnostic Properties of Protocol 2

5.3.1. Diagnostic Properties of Protocol 2: No State-Ambiguous Traces

The diagnostic properties of Protocol 2, if there are no state ambiguous traces in $L(G)$, are summarized by Theorem 5, whose proof is based on the following proposition.

PROPOSITION 3 *Let q_1, q_2, and q be the states of the diagnosers G_{d1}, G_{d2}, and G_d, respectively, after the system executed the trace $s = s_1 a$, where $a \in \Sigma_o$. If there are no state-ambiguous traces in $L(G)$, then we have the following:*

1. $q = q_1 \cap q_2$ *if* $a \in \Sigma_{o1} \cap \Sigma_{o2}$.
2. $q = q_1 \cap UR_2(q_2)$ *if* $a \in \Sigma_{o1} \backslash \Sigma_{o2}$.
3. $q = UR_1(q_1) \cap q_2$ *if* $a \in \Sigma_{o2} \backslash \Sigma_{o1}$.

Proof:

Proof of 1. We first note that

$$q \subseteq q_i, \quad i = \{1, 2\}. \tag{24}$$

The inclusion is true since the set of observable events Σ_{oi} is a subset of the original set of observable events Σ_o, and $a \in \Sigma_{o1} \cap \Sigma_{o2}$. From (24) we have that

$$q \subseteq q_1 \cap q_2. \tag{25}$$

Next we show that

$$q_1 \cap q_2 \subseteq q. \tag{26}$$

To prove (26) we proceed by contradiction. Assume that there exists $(x, l_x) \in Q_o$ such that

$$(x, l_x) \in q_1 \cap q_2 \text{ but } (x, l_x) \notin q. \tag{27}$$

By construction we know that $\delta(x_0, s) \neq x$ since if x were the true system state then we should have $(x, l_x) \in q$. The fact that $\delta(x_0, s) \neq x$ implies that there exists a state $y \in X$ such that $\delta(x_0, s) = y$ and (y, l_y) belongs to q, q_1 and q_2, where l_y is the failure label

associated with trace s. Therefore, there exist traces s', s'' in $L(G)$, s', s'' not necessarily distinct, such that

$$\begin{aligned} \delta(x_0, s') &= x, \\ \delta_{d1}(q_{01}, s') &= q_1, \\ P_1(s) &= P_1(s'), \\ \delta(x_0, s'') &= x, \\ \delta_{d2}(q_{02}, s'') &= q_2, \\ P_2(s) &= P_2(s''). \end{aligned} \tag{28}$$

Moreover, s', s'' share the same failure properties since the label associated with state x along s' and s'' is the same (and equal to l_x in our notation). In addition since $(x, l_x) \notin q$,

$$\delta_d(q_0, s') = q' \neq q, \quad \delta_d(q_0, s'') = q'' \neq q. \tag{29}$$

From Equation 29 it follows that

$$P(s) \neq P(s') \text{ and } P(s) \neq P(s''). \tag{30}$$

From (27), (30) and the fact that s', s'' share the same failure properties it follows that s is a state-ambiguous trace. Hence, by contradiction (26) is true. From (25) and (26) it follows that

$$q = q_1 \cap q_2. \tag{31}$$

Proof of 2. We first note that

$$q \subseteq q_1, \text{ and } q \subseteq UR_2(q_2). \tag{32}$$

The first inclusion is true as argued in the proof of *1*, and the second follows from the definition of the unobservable reach (cf. Definition 19) since $a \notin \Sigma_{o2}$. The remaining of the proof proceeds as in *1* with the minor change of replacing q_2 with $UR_2(q_2)$.

Proof of 3. We first note that

$$q \subseteq q_2, \text{ and } q \subseteq UR_1(q_1). \tag{33}$$

The first inclusion is true as argued in the proof of *1*, and the second follows from the definition of the unobservable reach (cf. Definition 19) since $a \notin \Sigma_{o1}$. The remaining of the proof proceeds as in *1* with the minor change of replacing q_1 with $UR_1(q_1)$. ■

Proposition 3 is used to prove the main result concerning the diagnostic properties of Protocol 2 if there are no state-ambiguous traces in $L(G)$.

THEOREM 5 *If there are no state-ambiguous traces in $L(G)$, we have the following:*
(i) The coordinator's diagnostic information C under Protocol 2 is the same as the state of the centralized diagnoser G_d.
(ii) Protocol 2 achieves the same diagnostic performance as the centralized diagnoser.

Proof: The proof of (i) is a direct consequence of Proposition 3 since every message is received in the order it was sent (by Assumptions **A4** and **A5**). From part (i) and the specification of the coordinator's decision rule, it follows that Protocol 2 achieves the same diagnostic performance as the centralized diagnoser. ■

A consequence of Proposition 3 and Theorem 5 is the ability to save on communication (by skipping messages) while maintaining the same diagnostic performance. This is explained in detail in Section 5.5.

5.3.2. Diagnostic Properties of Protocol 2: No Failure-Ambiguous Traces

In this section, we study the performance of Protocol 2 if there are no failure-ambiguous traces in $L(G)$. Based on the "failure-ambiguous traces" concept we have the following proposition.

PROPOSITION 4 *Let q_1, q_2, and q be the states of the diagnosers G_{d1}, G_{d2}, and G_d, respectively, after the system executed the trace $s = s_1 a$, where $a \in \Sigma_o$. If there are no failure-ambiguous traces (with respect to failure type F_i) in $L(G)$, then we have the following:*

1. *If $a \in \Sigma_{o1} \cap \Sigma_{o2}$ then q is F_i-certain if and only if $q_1 \cap q_2$ is F_i-certain.*
2. *If $a \in \Sigma_{o1} \backslash \Sigma_{o2}$ then q is F_i-certain if and only if $q_1 \cap UR_2(q_2)$ is F_i-certain.*
3. *If $a \in \Sigma_{o2} \backslash \Sigma_{o1}$ then q is F_i-certain if and only if $UR_1(q_1) \cap q_2$ is F_i-certain.*

Proof:

Proof of 1. From (25) we have that q is F_i-certain if $q_1 \cap q_2$ is F_i-certain. We need to prove that q is F_i-certain only if $q_1 \cap q_2$ is F_i-certain. We proceed by contradiction. Assume that q is F_i-certain but $q_1 \cap q_2$ is not. Assume that $\delta(x_0, s) = x$. By construction there exists $(x, l_x), (y, l_y) \in Q_o$, $F_i \in l_x$, $F_i \notin l_y$ such that

$$(x, l_x), (y, l_y) \in q_1 \cap q_2 \text{ but } (y, l_y) \notin q. \tag{34}$$

Therefore, there exist traces s', s'' in $L(G)$, s', s'' not necessarily distinct, such that

$$\begin{aligned} \delta(x_0, s') &= y, \\ \delta_{d1}(q_{01}, s') &= q_1, \\ P_1(s) &= P_1(s'), \end{aligned}$$

$$\begin{aligned} \delta(x_0, s'') &= y, \\ \delta_{d2}(q_{02}, s'') &= q_2, \\ P_2(s) &= P_2(s''). \end{aligned} \tag{35}$$

Moreover, s', s'' share the same failure properties since the label associated with state y along s' and s'' is the same (and equal to l_y in our notation). In addition since we know that q is F_i-certain and $F_i \notin l_y$, then s, s' and s, s'' do not share the failure property F_i. Moreover, since $(y, l_y) \notin q$ we have that

$$\delta_d(q_0, s') = q' \neq q, \quad \delta_d(q_0, s'') = q'' \neq q. \tag{36}$$

From (36) it follows that

$$P(s) \neq P(s') \text{ and } P(s) \neq P(s''). \tag{37}$$

From (34), (37) and the facts that s', s'' share the same failure properties and s, s' and s, s'' do not share the failure type F_i it follows that s is a failure-ambiguous trace (with respect to failure type F_i). Hence, by contradiction q is F_i-certain only if $q_1 \cap q_2$ is F_i-certain.

Proof of 2. From (32) we have that q is F_i-certain if $q_1 \cap UR_2(q_2)$ is F_i-certain. We need to prove that q is F_i-certain only if $q_1 \cap UR_2(q_2)$ is F_i-certain. The proof proceeds as in *1* with the minor change of replacing q_2 with $UR_2(q_2)$.

Proof of 3. From (33) we have that q is F_i-certain if $UR_1(q_1) \cap q_2$ is F_i-certain. We need to prove that q is F_i-certain only if $UR_1(q_1) \cap q_2$ is F_i-certain. The proof proceeds as in *1* with the minor change of replacing q_1 with $UR_1(q_1)$. ■

Proposition 4 is used to prove the main result concerning the diagnostic properties of Protocol 2 if there are no failure-ambiguous traces in $L(G)$.

THEOREM 6 *If there are no failure-ambiguous traces in $L(G)$, Protocol 2 achieves the same diagnostic performance as the centralized diagnoser.*

Proof: The proof is a direct consequence of Proposition 4 (since every message is received in the order it was sent by Assumptions **A4** and **A5**) and the specification of the coordinator's decision rule it. ■

5.3.3. Discussion

Theorem 5*(i)* states that the coordinator's diagnostic information is equal to the centralized diagnoser state if there are no state-ambiguous traces in $L(G)$. However, for the purpose of failure diagnosis this is not necessary. To diagnose all failure types the coordinator's diagnostic information should be F_i-certain (after the occurrence of a failure of type F_i) if and only if the centralized diagnoser state is F_i-certain, without the need of having the two entities equal. This has been established if there are no failure-ambiguous traces in $L(G)$.

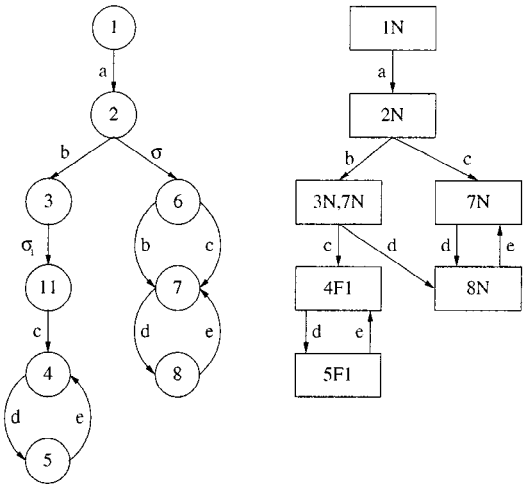

Figure 6. The system (left) and centralized diagnoser for Example 11.

Since absence of failure-ambiguous traces is less restrictive than absence of state-ambiguous traces, the result of Theorem 6 is stronger than that of Theorem 5.

Although for the purpose of failure diagnosis it is more appropriate to study the performance of Protocol 2 if there are no failure-ambiguous traces in $L(G)$, Theorem 5 is used to compare the performance of Protocol 2 with that of Protocol 1. Theorem 3 states that irrespective of the system structure, Protocol 1 reconstructs the state of the centralized diagnoser and hence performs as well as the centralized diagnoser. Therefore, the performance of Protocol 2 is inferior to that of Protocol 1 since reconstructing the state of the centralized diagnoser and performing as well as the centralized diagnoser is conditioned on the fact that there are no state-ambiguous traces.

The following example shows that if the system contains state-ambiguous or failure-ambiguous traces the results of Propositions 3 and 4 are not, in general, true.

Example 11. Consider the system shown in Figure 6. The set of events is $\Sigma = \{a, b, c, d, e, \sigma, \sigma_1\}$, $\Sigma_{uo} = \{\sigma, \sigma_1\}$, $\Sigma_{o1} = \{a, b, d\}$, $\Sigma_{o2} = \{a, c, e\}$ and $\Sigma_{f1} = \{\sigma_1\}$. The trace $s = ab\sigma_1 c(de)^*$ is state/failure-ambiguous since (1) $P_1(s) = P_1(a\sigma b(de)^*)$ but $P(s) \neq P(a\sigma b(de)^*)$, (2) $P_2(s) = P_2(a\sigma c(de)^*)$ but $P(s) \neq P(a\sigma c(de)^*)$, (3) $a\sigma b(de)^*$ and $a\sigma c(de)^*$ share the same failure properties (here both traces do not have failure events), (4) s has a failure of type F_1 while $a\sigma b(de)^*$ and $a\sigma c(de)^*$ exhibit only normal behavior. The diagnosers G_{d1} and G_{d2} are shown in Figure 7. If the system executes the trace $ab\sigma_1 c(de)^*$ then after the occurrence of the event d, R_1 = (5F1,8N) and R_4 = (4F1,7N,5F1,8N). Therefore, by applying action **DR1** (since d is observed by site 1 only) the coordinator's diagnostic information C is equal to $R_1 \cap R_4 = q_1 \cap UR_2(q_2) = (5F1, 8N)$. Along the same trace, after the occurrence of event d the centralized diagnoser state is $q = 5F1$ which is neither equal to C nor has the same diagnostic properties as C. Thus

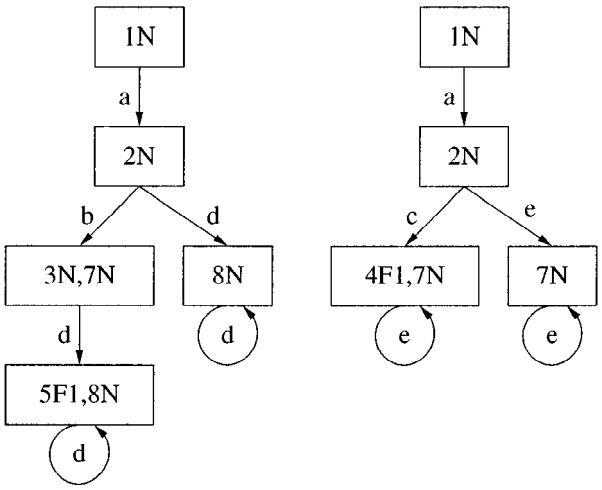

Figure 7. The diagnosers G_{d1} (left) and G_{d2} for Example 11.

the presence of state/failure-ambiguous traces shows that the results of Propositions 3 and 4 are not, in general, true.

The next example shows that the "state-ambiguous trace" ("failure-ambiguous trace") condition is not necessary for Proposition 3 (respectively 4) to be true.

Example 12. Consider the system shown in Figure 8. The set of events is $\Sigma = \{a, b, c, d, e, \sigma, \sigma_1\}$, $\Sigma_{uo} = \{\sigma, \sigma_1\}$, $\Sigma_{o1} = \{a, b, d\}$, $\Sigma_{o2} = \{a, c, e\}$ and $\Sigma_{f1} = \{\sigma_1\}$. The trace $s = ab\sigma_1 c(de)^*$ is state/failure-ambiguous since (1) $P_1(s) = P_1(a\sigma b(de)^*)$ but $P(s) \neq P(a\sigma b(de)^*)$, (2) $P_2(s) = P_2(a\sigma c(de)^*)$ but $P(s) \neq P(a\sigma c(de)^*)$, (3) $a\sigma b(de)^*$ and $a\sigma c(de)^*$ share the same failure properties (here both traces do not have failure events), (4) s has a failure of type F_1 while $a\sigma b(de)^*$ and $a\sigma c(de)^*$ exhibit only normal behavior. The diagnosers G_{d1} and G_{d2} are shown in Figure 9. If the system executes the trace $ab\sigma_1 c(de)^*$ then after the occurrence of the event d, $R_1 = (5F1, 8N)$ and $R_4 = (4F1, 9N, 5F1, 10N)$. Therefore, by applying action **DR1** (since d is observed by site 1 only) the coordinator's diagnostic information C is equal to $R_1 \cap R_4 = q_1 \cap U R_2(q_2) = 5F1$. Along the same trace, after the occurrence of event d the centralized diagnoser state is $q = 5F1 = q_1 \cap U R_2(q_2)$ and by definition it has the same diagnostic properties as C. Consequently, the condition on state/failure-ambiguous traces in Propositions 3 and 4 is only sufficient for these propositions to be true.

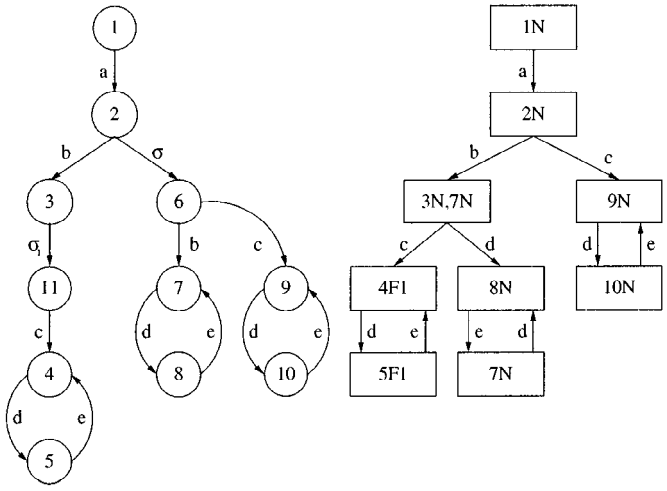

Figure 8. The system (left) and centralized diagnoser for Example 12.

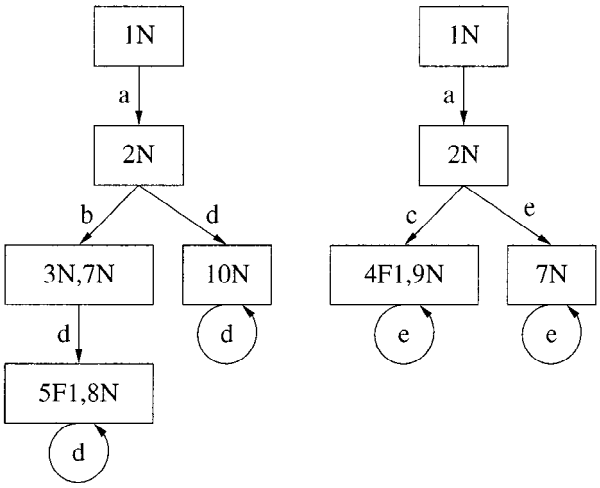

Figure 9. The diagnosers G_{d1} (left) and G_{d2} for Example 12.

5.4. Necessary and Sufficient Conditions for Diagnosability

The results of Section 5.3 imply that if there are no failure-ambiguous traces in $L(G)$, the necessary and sufficient conditions for diagnosability with respect to Protocol 2 can be stated with respect to the centralized diagnoser as follows.

THEOREM 7 *If there are no failure-ambiguous traces in $L(G)$, a live and prefix-closed language L is diagnosable with respect to Protocol 2, the set of projections P_1, P_2 and the failure partition Π_f on Σ_f if and only if the diagnoser G_d does not have F_i-indeterminate cycles for all failure types F_i.*

Proof: Sufficiency($\Leftarrow$). Suppose G_d does not have F_i-indeterminate cycles. Consider a trace $st \in L(G)$ such that $s \in \Psi(\Sigma_{fi})$, and t is long enough, i.e., $\|t\| > n$, where n can be arbitrarily large. Then, by assumption, st', $t' \in \bar{t}$ does not lead to an F_i-indeterminate cycle in G_d. Consequently, G_d will enter an F_i-certain state within a finite number of steps, say n_i'. This implies by Theorem 6 that the coordinator's diagnostic information C will be F_i-certain within a finite number of steps equal to n_i'. Therefore, $L(G)$ is diagnosable with respect to Protocol 2.

Necessity($\Rightarrow$). We prove the contrapositive. Assume that G_d does enter an F_i-indeterminate cycle. This implies that the coordinator's diagnostic information C, which carries the same diagnostic properties as the centralized diagnoser state by Theorem 6, will remain F_i-uncertain for an arbitrarily long number of steps. Hence, L is not diagnosable with respect to Protocol 2. ■

Having checked that the diagnoser G_d is capable of diagnosing all failure types, the next step could be to find a test to check whether failure-ambiguous traces exist or not. We bypass this and instead provide a direct test to verify whether Protocol 2 performs as well as the centralized diagnoser. If the test fails we conclude that there are failure-ambiguous traces in the language. Otherwise, we know that Protocol 2 diagnoses all failure types that are diagnosed by the centralized diagnoser; however we cannot claim that there are no failure-ambiguous traces in the language. In order to present this test, we introduce G_{test2}. G_{test2} is the FSM

$$G_{test2} = (Q_g, \Sigma_o, \delta_g, g_0) \tag{38}$$

where the state space $Q_g \subseteq Q_{d1} \times Q_{d2} \times Q_d \times (Q_{d1} \cap Q_{d2})$, Σ_o is the set of events of G_{test2} and $g_0 = (q_{01}; q_{02}; q_0; q_{01} \cap q_{02}) = (\{(x_0, \{N\})\}; \{(x_0, \{N\})\}; \{(x_0, \{N\})\}; \{(x_0, \{N\})\})$ is the initial state of G_{test2}. A state p in G_{test2} is denoted by $(g_1; g_2; g; g_c)$ where $g_1 \in Q_{d1}$, $g_2 \in Q_{d2}$, $g \in Q_d$ and $g_c \in Q_{d1} \cap Q_{d2}$. The partial transition function δ_g is defined as follows:

$$\delta_g((g_1; g_2; g; g_c), \sigma) = \begin{cases} (\delta_{d1}(g_1, \sigma); \delta_{d2}(g_2, \sigma); \delta_d(g, \sigma); \delta_{d1}(g_1, \sigma) \cap \delta_{d2}(g_2, \sigma)) \\ \text{if } \sigma \in \Sigma_{o1} \cap \Sigma_{o2} \\ (\delta_{d1}(g_1, \sigma); g_2; \delta_d(g, \sigma); \delta_{d1}(g_1, \sigma) \cap UR_2(g_2)) \\ \text{if } \sigma \in \Sigma_{o1} \backslash \Sigma_{o2} \\ (g_1; \delta_{d2}(g_2, \sigma); \delta_d(g, \sigma); UR_1(g_1) \cap \delta_{d2}(g_2, \sigma)) \\ \text{if } \sigma \in \Sigma_{o2} \backslash \Sigma_{o1}. \end{cases}$$

The ideas and objective behind introducing this machine are the following:

1. Synchronize the operation of the diagnosers G_{d1} and G_{d2} to be able to generate C, the coordinator's diagnostic information.

2. Make sure that the synchronized behavior is indeed a legal observed behavior of the system.

It can be verified that $L(G_{test2}) = L(G_d)$. Therefore, G_{test2} observes the system behavior as would G_d after the execution of a given trace s, provides information about the states of the diagnosers G_{d1} and G_{d2} after the execution of s, and computes the coordinator's diagnostic information, C. We are interested in detecting simultaneous occurrences of F_i-indeterminate cycles in both G_{d1} and G_{d2} because by comparing the coordinator's diagnostic information with the centralized diagnoser's information along these cycles we may be able to determine whether Protocol 2 performs as well as a centralized diagnoser. To precisely describe how we do so, we need the following definitions.

Definition 20. A state $p = (g_1; g_2; g; g_c)$ in G_{test2} is said to be ambiguous if g is not normal and g, g_c do not have the same failure properties. We say that g, g_c have the same failure properties when the following is true: g is F_i-certain (uncertain) if and only if g_c is F_i-certain (uncertain), for all failure types F_i.

Definition 21. A cycle in G_{test2} is said to be F_i-indeterminate if the corresponding cycles in G_{d1} and G_{d2} are both F_i-indeterminate.

Definition 22. A cycle in G_{test2} is said to be F_i-ambiguous if it is F_i-indeterminate **and** all its states are ambiguous.

The notion of an ambiguous cycle is helpful in providing conditions under which the language is diagnosable as the following theorem indicates.

THEOREM 8 *(i) A live and prefix-closed language L is diagnosable with respect to Protocol 2, the set of projections P_1, P_2 and the failure partition Π_f on Σ_f if for all failure types F_i, the following conditions are true: (a) G_{test2} does not have F_i-ambiguous cycles* **and** *(b) G_d does not have F_i-indeterminate cycles.*

(ii) A live and prefix-closed language L modeled by the FSM G is diagnosable with respect to Protocol 2, the set of projections P_1, P_2 and the failure partition Π_f on Σ_f only if for all failure types F_i the following conditions are true: (a) G_{test2} corresponding to G does not have any F_i-ambiguous cycles **and** *(b) G_d does not have any F_i-indeterminate cycles.*

Proof: (i) Suppose G_{test2} does not have any F_i-ambiguous cycles and G_d does not have F_i-indeterminate cycles for all failure types F_i. Consider a trace $st \in L(G)$ such that $s \in \Psi(\Sigma_{fi})$ and t is long enough, i.e., $\|t\| > n$, where n can be arbitrarily large. By the construction of G_{test2} and the assumption that G_{test2} does not have any ambiguous cycles, st cannot lead to F_i-indeterminate cycles in both G_{d1} and G_{d2} along which the coordinator's diagnostic information does not have the same failure properties as the centralized diagnoser state. Therefore C cannot remain F_i-uncertain indefinitely since G_d does not have F_i-indeterminate cycles; hence $L(G)$ is diagnosable with respect to Protocol 2.

(ii) Assume $L(G)$ is diagnosable with respect to Protocol 2 and is modeled by the FSM G. Consider a trace $st \in L(G)$ such that $s \in \Psi(\Sigma_{fi})$, t is long enough, i.e., $\|t\| > n$, where n can be arbitrarily large, and s leads to F_i-indeterminate cycles in both G_{d1} and

G_{d2}. By assumption we know that the coordinator's diagnostic information C will be F_i-certain in a finite number of steps along st. When C is F_i-certain then the corresponding centralized diagnoser state is F_i-certain because of the rule generating C (cf. Section 5.2.3 and (25), (32) and (33)). Therefore, following st the centralized diagnoser does not enter an F_i-indeterminate cycle and G_{test2} does not enter an F_i-ambiguous cycle (because the coordinator's diagnostic information and the centralized diagnoser will both be F_i-certain along st within a finite number of steps). Since st is arbitrary, then for all failure types F_i, G_{test2} corresponding to G does not have any F_i-ambiguous cycles and G_d does not have any F_i-indeterminate cycles. ■

We note here that the necessary and sufficient conditions in Theorem 8 are almost identical, except that the necessary conditions depend on the machine representation of the language. We clarify this subtlety as follows. Protocol 2 is a state-based (as opposed to language-based) diagnostic scheme (cf. the definition of the decision rule in Section 5.2.3). Hence, we expect the necessary and sufficient conditions to depend on the FSM G that models the language L. In contrast, the necessary and sufficient conditions for diagnosability in the centralized case (cf. Theorem 1) do not depend on the FSM G, since the arguments used in the proof of Theorem 2 in Sampath et al. (1995) are all trace-based arguments, i.e., the diagnostic scheme is indeed language-based. In the case of Protocol 2, if G_{test2} does not have F_i-ambiguous cycles, and G_d does not have F_i-indeterminate cycles for all failure types F_i, then Protocol 2 diagnoses all failure types, and we need not worry about G in the statement of the sufficient conditions. However, if G_{test2} has F_i-ambiguous cycles, then we cannot assert that Protocol 2 cannot diagnose a failure of type F_i since there may exist another FSM G' of L, such that Protocol 2 diagnoses all failure types when it is combined with G'. Consequently, the necessary conditions indeed depend on G. The following examples illustrate the above discussion.

Example 13. Consider the system discussed in Example 11 and shown in Figure 6. G_{test2} is shown in Figure 10. The cycle labeled A in the Figure is a F_1-ambiguous cycle: it can be verified that the cycles $(5F1, 8N)$ and $(4F1, 7N)$ are F_1-indeterminate cycles in G_{d1} and G_{d2}, respectively, and both cycles are ambiguous since the state of the centralized diagnoser is F_1-certain ($5F1$ or $4F1$) and the coordinator's diagnostic information is F_1-uncertain ($(5F1, 8N)$ or $(4F1, 7N)$). Therefore the system with the FSM representation shown in Figure 6 is not diagnosable under Protocol 2.

Example 14. Consider the system discussed in Example 12 and shown in Figure 8. The systems of Examples 11 and 12 provide two different FSM representations of the same language. G_{test2} for the system of Example 12 is shown in Figure 11. The cycle labeled A in the figure is a F_1-indeterminate cycle: it can be verified that the cycles $(5F1, 8N)$ and $(4F1, 9N)$ are F_1-indeterminate cycles in G_{d1} and G_{d2}, respectively; however the states $(5F1, 8N : 4F1, 9N : 5F1 : 5F1)$ and $(5F1, 8N : 4F1, 9N : 4F1 : 4F1)$ share the same failure properties (F_1-certain), therefore the cycle is not F_1-ambiguous. Hence by Theorem 8 the system with the FSM representation shown in Figure 8 is diagnosable under Protocol 2. Note here that although the language exhibits failure-ambiguous traces

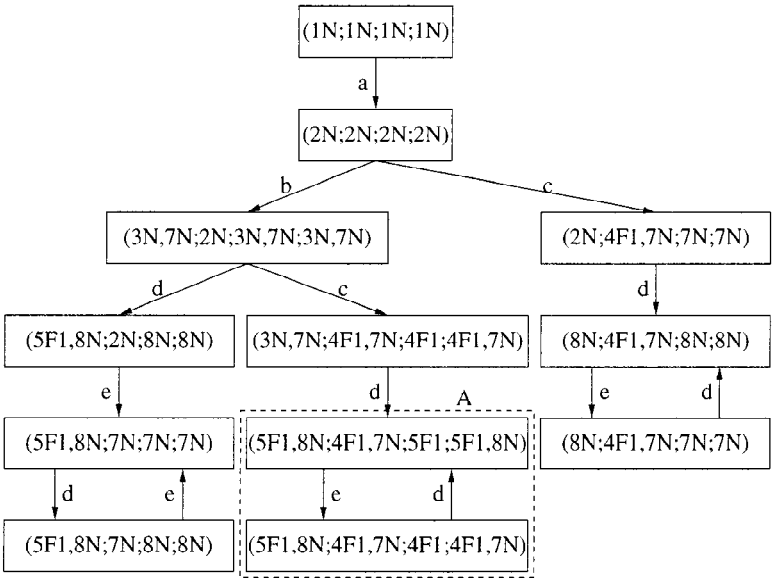

Figure 10. G_{test2} for Example 13.

(cf. Example 12), the system is diagnosable since we are able to verify using G_{test2} that Protocol 2 performs as well as the centralized diagnoser based on G shown in Figure 8. In Example 13 we showed that the same language represented by another G (cf. Figure 6) is not diagnosable under Protocol 2, hence proving that the system model G should be taken into consideration in the necessity proof of Theorem 8.

5.5. Discussion

We first note that the performance of Protocol 2 is inferior to that of Protocol 1 because only under the restrictions on the system structure discussed in Section 5.3, Protocol 2 performs as well as the centralized diagnoser. However, the communication, memory, and processing requirements for Protocol 2 are less than those of Protocol 1. Indeed, the generation of diagnostic information at the local sites, in case of Protocol 2, requires less time and memory than the generation of diagnostic information in Protocol 1. Furthermore less information per observed event has to be communicated to the coordinator under Protocol 2. The coordinator's information update and decision rules for Protocol 2 are simpler to implement than the ones used for Protocol 1.

The partitioning of observable events is crucial in deciding whether a language is diagnosable under Protocol 2 or not. In fact, failure-ambiguous traces, which may force Protocol 2 not to perform as well as the centralized diagnoser, are defined with respect to the projections P_1 and P_2 (cf. Definition 18). Therefore, changing the partitions P_1

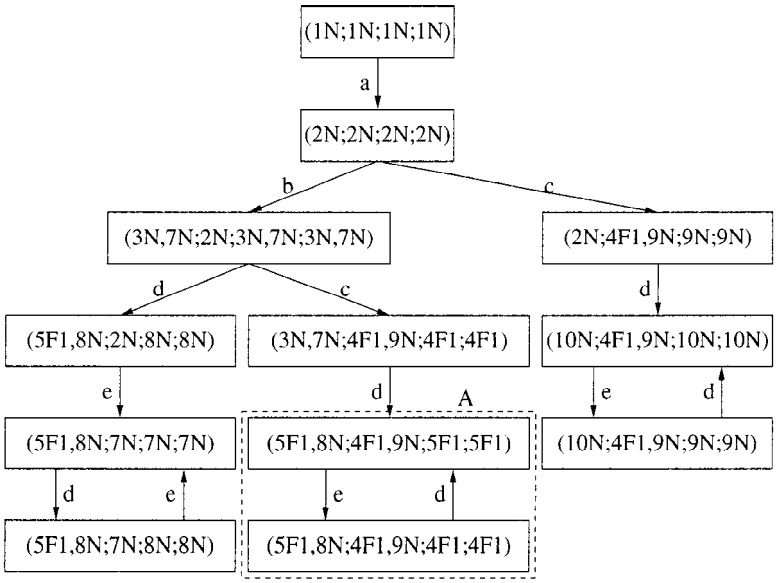

Figure 11. G_{test2} for Example 14.

and P_2 may eliminate the presence of failure-ambiguous traces, hence allowing Protocol 2 to perform as well as the centralized diagnoser. The following example illustrates this idea.

Example 15. Consider the system of Example 1 shown in Figure 2 with $\Sigma = \{a, b, c, d, e, \sigma\}$, $\Sigma_{uo} = \{\sigma\}$, $\Sigma_{f1} = \{\sigma\}$, $\Sigma_{o1} = \{a, c, d, e\}$, and $\Sigma_{o2} = \{b, d, e\}$. In Example 9 we showed that the trace $bac\sigma(de)^*$ is failure-ambiguous, hence Protocol 2 may not perform as well as the centralized diagnoser with the partitions P_1 and P_2. Indeed, this can be seen by checking G_{test2} for the above system (cf. Figure 12): the cycle labeled A in the figure is F_1-ambiguous because the cycles $\{(8N, 10F1), (6N, 9F1)\}$ and $\{(8N, 10F1, 11N), (6N, 9F1, 12N)\}$ are F_1-indeterminate in G_{d1} and G_{d2}, respectively, and the correponding centralized diagnoser state is F_1-certain ($10F1$ and $9F1$) while the coordinator's diagnostic information is F_1-uncertain ($(8N, 10F1)$ and $(6N, 9F1)$). If we consider now a new partitioning of the observable events where Σ_{o1} is as before and $\Sigma_{o2} = \{a, b, d, e\}$, then Protocol 2 performs as well as the centralized diagnoser: G_{test2} for the system with the new partitioning of observable events (i.e. the new set of projections) is shown in Figure 13; clearly there are no F_i-ambiguous cycles in G_{test2}.

It is worth noting that, if there are no state-ambiguous traces in $L(G)$, the information state available at the coordinator's site is sufficient, at any instant of time, to obtain the estimate of the centralized diagnoser. Now we argue that, if there are no state-

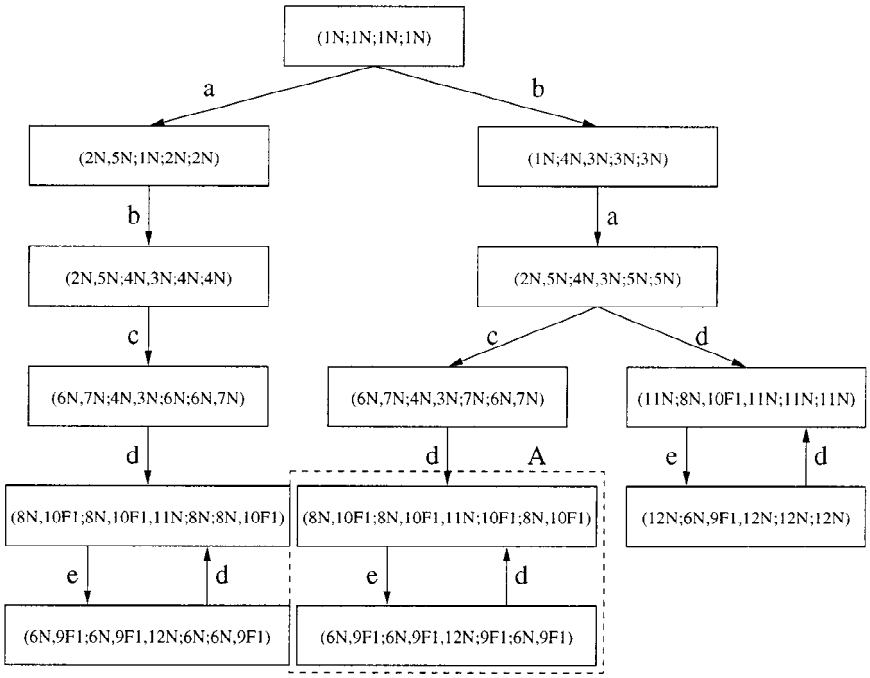

Figure 12. G_{test2} for Example 15.

ambiguous traces in $L(G)$, Protocol 2 can achieve the same performance as the centralized diagnoser *even when there is no continuous communication between the local sites and the coordinator*. This can be done as follows: communication is initiated by a request from the coordinator rather than the occurrence of an observable event. We assume that the request from the coordinator reaches the two sites simultaneously or at least no observable event is executed from the time the request reaches one site until it reaches the other site. Upon receiving the request each site communicates to the coordinator its current state and unobservable reach and a status bit (as defined earlier) specifying whether the last event it observed was common or not. In this case, we show in [8] that by slightly modifying the coordinator's decision rule described earlier, Theorems 5 and 6 still hold. Thus communication is reduced while achieving the same performance. Situations where savings in communication are of paramount importance arise in networks where the nodes (sites) are low energy battery-powered mobile units.

The feature of Protocol 2 discussed above is also present in decentralized estimation of linear Gaussian systems, Speyer (1979) and Willsky et al. (1982). However, in Speyer (1979) and Willsky et al. (1982) there are no restrictions on the structure of the linear Gaussian system. On the other hand, the estimation problems in Speyer (1979) and Willsky et al. (1982) are linear, whereas Protocol 2 deals with a nonlinear estimation problem. De-

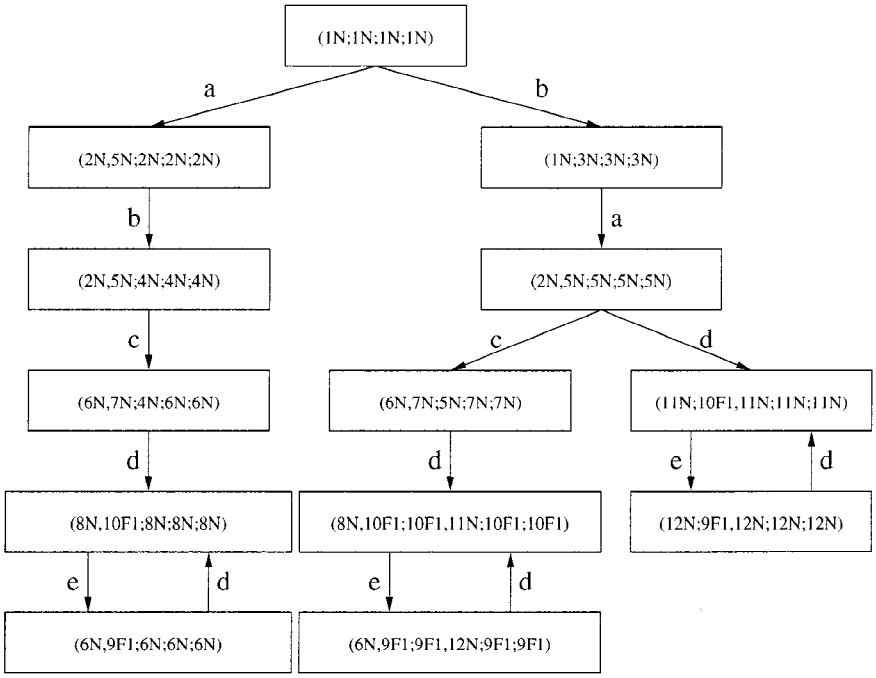

Figure 13. G_{test2} with the new set of projections for Example 15.

centralized nonlinear estimation problems for stochastic systems have been investigated in Castanon and Teneketzis (1985). The coordinated decentralized estimation protocol proposed in Castanon and Teneketzis (1985) achieves the same performance as the optimal centralized estimator under no restrictions on the system structure, but requires continuous communication between the local sites and the coordinator and assumes that the coordinator has knowledge of the system structure. The above comparison shows that, under the assumptions on the system structure discussed in Section 5.1, Protocol 2 has some remarkable features.

Finally, Example 14 shows that although the local diagnosers remain F_i-uncertain indefinitely after the occurrence of a failure of type F_i, Protocol 2 detects and isolates the failure. The process of identifying the failures is achieved through the decision rule implemented at the coordinator. The decision rule identifies failures based on the diagnostic information C. The information update rule (cf. Section 5.2.3) processes the messages received from the local sites to update the diagnostic information C. In some practical cases, the coordinator cannot wait to process the data to make decisions, rather it should declare the occurrence of failures based on the raw information it receives from the local sites. The next section introduces a protocol that addresses this issue.

6. A Third Coordinated Decentralized Protocol: Protocol 3

6.1. Objective and Assumptions

In this section, we present a protocol where the coordinator declares the occurrence of failures based on the raw information it receives from the local sites. We call this protocol Protocol 3.

To analyze the performance of Protocol 3 we introduce the notion of fully-ambiguous traces with respect to the projections P_1 and P_2.

Definition 23. A trace $s \in L(G)$ is said to be fully-ambiguous with respect to the projections P_1 and P_2 and the failure type F_i if there exist two traces, s' and s'' in $L(G)$ such that s' and s'' are arbitrarily long, not necessarily distinct, and the following is true:

1. $P_1(s) = P_1(s')$ but $P(s) \neq P(s')$.

2. $P_2(s) = P_2(s'')$ but $P(s) \neq P(s'')$.

3a. $F_i \in s$ but $F_i \notin s'$.

3b. $F_i \in s$ but $F_i \notin s''$.

This definition says that the traces s, s' and s, s'' can be dinstinguished under the projection P; however s and s' are not distinguishable under P_1 while s and s'' are not distinguishable under P_2. Furthermore, there is a difference in failure properties between s, s' and s, s'': if F_i belongs to s then it does not belong to neither s' nor s'' or vice-versa. Note here that a failure-ambiguous trace is a fully-ambiguous trace; however the reverse is not true since there are no restrictions on the failure properties of s' and s'' in the definition of a fully-ambiguous trace. Thereafter when we refer to a trace as being fully-ambiguous, the projections P_1 and P_2 and the failure type F_i will be understood from the context. The concept of "fully-ambiguous traces" is illustrated by the following example.

Example 16. Consider the system discussed in Example 12 and shown in Figure 8. The set of events is $\Sigma = \{a, b, c, d, e, \sigma, \sigma_1\}$, $\Sigma_{uo} = \{\sigma, \sigma_1\}$, $\Sigma_{f1} = \{\sigma_1\}$, $\Sigma_{o1} = \{a, b, d\}$ and $\Sigma_{o2} = \{a, c, e\}$. The trace $s = ab\sigma_1 c(de)^*$ is fully-ambiguous (with respect to failure type F_1) since (1) $P_1(s) = P_1(a\sigma b(de)^*) = ab(d)^*$ but $P(s) = abc(de)^* \neq ab(de)^* = P(a\sigma b(de)^*)$, (2) $P_2(s) = P_2(a\sigma c(de)^*) = ac(e)^*$ but $P(s) = abc(de)^* \neq ac(de)^* = P(a\sigma c(de)^*)$ and (3) $F_1 \in s$, but $F_1 \notin a\sigma b(de)^*$ and $F_1 \notin a\sigma c(de)^*$. Note here that we could have concluded that s is fully-ambiguous since we showed in Example 12 that s is failure-ambiguous.

We will study the performance of Protocol 3 under Assumptions **A1**–**A8** (cf. Section 3.1) and the following additional assumption.

A10 There are no fully-ambiguous traces (with respect to all failure types) in $L(G)$.

6.2. Specification of the Protocol

In this section we specify Protocol 3 in detail. We begin by discussing the diagnostic information at the local sites.

6.2.1. Diagnostic Information at Local Sites

We implement diagnosers at the local sites. Therefore, the state of the diagnoser, after the occurrence of an observable event, is the diagnostic information based on which the site is supposed to infer the occurrence of failures.

6.2.2. Communication Rules

Since the coordinator is supposed to declare the occurrence of failures based on the raw information provided by the local sites, the information communicated from the local sites should be as simple and concise as possible. Consequently we define the following communication rules:

- **[CRi]**, $i = 1, 2$: After the agent at site i observes an event $\sigma \in \Sigma_{oi}$ that leads to an F_i-certain state in the diagnoser G_{di}, it communicates the label F_i to the coordinator, meaning that a failure of type F_i has occurred.

6.2.3. Decision Rule

The coordinator declares that a failure of type F_i has occurred once its diagnostic information C is F_i-certain. Since local sites communicate the failure labels detected by their corresponding diagnoser, once the coordinator receives a message, either from site 1 or site 2, containing the information F_i, it declares the occurrence of a failure of type F_i and broadcasts the information to the failure recovery module.

The coordinator's diagnostic information C is updated each time a message is received at its site. The rule update is $C = C \cup \{F_i\}$ where F_i is the last incoming message. For the sake of consistency with the definition of diagnosability (cf. Definition 8), when reading that C is F_i-certain, we understand that F_i belongs to C.

6.3. Diagnostic Properties of Protocol 3

We first present a lemma that relates the existence of F_i-indeterminate cycles in the local diagnoser to fully-ambiguous traces (with respect to failure type F_i).

LEMMA 2 *Consider a trace s in $L(G) \cap \Psi(\Sigma_{fi})$. The trace s is fully-ambiguous with respect to failure type F_i if and only if s leads to F_i-indeterminate cycles in G_{d1} and G_{d2}.*

Proof: Sufficiency ($\Leftarrow$). Assume that s leads to F_i-indeterminate cycles in G_{d1} and G_{d2}. By definition (cf. Definition 5), there exist traces s' and s'' in $L(G)$ arbitrarily long such that the following is true:

$$P_1(s) = P_1(s'),\ F_i \in s \text{ but } F_i \notin s', \tag{39}$$

and

$$P_2(s) = P_2(s''),\ F_i \in s \text{ but } F_i \notin s''. \tag{40}$$

(39) and (40) imply that s is fully-ambiguous with respect to F_i.

Necessity ($\Rightarrow$). The fact that s is fully ambiguous with respect to F_i implies that there exist two traces s' and s'' such that

$$P_1(s) = P_1(s'),\ F_i \in s \text{ but } F_i \notin s', \tag{41}$$

and

$$P_2(s) = P_2(s''),\ F_i \in s \text{ but } F_i \notin s''. \tag{42}$$

(41) and (42) imply that there exists an F_i-indeterminate cycle in G_{d1} and G_{d2} simultaneously, i.e., following the trace s. ■

The above lemma establishes the fact that following the execution of a trace which is not fully-ambiguous, the local diagnosers G_{d1} and G_{d2} cannot loop "simultaneously" in indeterminate cycles. By "simultaneously" we understand following the execution of a trace in the system.

The diagnostic properties of Protocol 3 are summarized by the following theorem.

THEOREM 9 *Protocol 3 performs as well as the centralized diagnoser if and only if there are no fully-ambiguous traces (with respect to all failure types) in the language.*

Proof: Sufficiency($\Leftarrow$). Assume that there are no fully-ambiguous traces in the language. Consider a trace $st \in L(G)$ such that $s \in \Psi(\Sigma_{fi})$, t is long enough, i.e., $\|t\| > n$, where n can be arbitrarily large and st leads to an F_i-certain state in the centralized diagnoser. By the implication of Lemma 2 s cannot lead to F_i-indeterminate cycles in both G_{d1} and G_{d2}. Therefore the state of G_{d1} or that of G_{d2} will be F_i-certain in a finite number of steps, which implies that either G_{d1} or G_{d2} will diagnose the failure. Since s is arbitrary, all failures diagnosed by the centralized diagnoser are diagnosed under Protocol 3. Therefore, Protocol 3 performs as well as the centralized diagnoser.

Necessity($\Rightarrow$). We prove the contrapositive. Assume that there are fully-ambiguous traces in the language. Consider a fully-ambiguous trace $st \in L(G)$ such that $s \in \Psi(\Sigma_{fi})$, t is long enough, i.e., $\|t\| > n$, where n can be arbitrarily large and st leads to a F_i-certain state in the centralized diagnoser. By Lemma 2, we have that s leads to F_i-indeterminate cycles in both G_{d1} and G_{d2}. Therefore the state of G_{d1} and that of G_{d2} will be F_i-uncertain indefinitely (along st), which implies that $F_i \notin C$. Therefore, L is not diagnosable under Protocol 3, hence Protocol 3 do not perform as well as the centralized diagnoser. ■

6.4. Necessary and Sufficient Conditions for Diagnosability

The results of Section 6.3 imply that if there are no fully-ambiguous traces in $L(G)$, the necessary and sufficient conditions for diagnosability with respect to Protocol 3 can be stated with respect to the centralized diagnoser as follows:

THEOREM 10 *If there are no fully-ambiguous traces in $L(G)$, a live and prefix-closed language L is diagnosable with respect to Protocol 3, the set of projections P_1, P_2 and the failure partition Π_f on Σ_f if and only if the diagnoser G_d does not have F_i-indeterminate cycles for all failure types F_i.*

Proof: The proof is a direct consequence of Theorem 9. ■

Having checked that the diagnoser G_d is capable of diagnosing all failure types, the next step could be to find a test to check whether fully-ambiguous traces exist or not. In order to present this test, we introduce G_{test3}. G_{test3} of the system G is defined as follows:

$$G_{test3} = G_{d1} \| G_{d2} \| G_d$$

where G_{d1}, G_{d2}, and G_d are as defined earlier. The following format for a state p of G_{test3} is adopted: $p = (q_1; q_2; q)$, where q_1, q_2, and q belong to Q_{d1}, Q_{d2}, and Q_d, respectively.

The ideas and objectives behind introducing this machine are the following:

1. Synchronize the operation of the diagnosers G_{d1} and G_{d2}. This necessitates their parallel composition.
2. Make sure that the synchronized behavior is indeed a legal observed behavior of the system. This necessitates the composition of $G_{d1} \| G_{d2}$ with the system diagnoser G_d.

Now we have that

$$L(G_{test3}) \stackrel{\triangle}{=} P_1^{-1}[L(G_{d1})] \cap P_2^{-1}[L(G_{d2})] \cap L(G_d) \tag{43}$$

where P_i^{-1}, $i = 1, 2$, is with respect to $\Sigma_{o1} \cup \Sigma_{o2}$ and not Σ. This fact implies that

$$L(G_{test3}) = L(G_d). \tag{44}$$

Therefore, G_{test3} observes the system behavior as would G_d after the execution of a given trace s, and also provides information about the states of the diagnosers G_{d1} and G_{d2} after the execution of s. Hence, using G_{test3}, it is possible to identify the states of the diagnosers G_{d1} and G_{d2} after the system has executed a trace in the language. We are interested in detecting simultaneous occurrences of F_i-indeterminate cycles in both G_{d1} and G_{d2}, since testing these cycles may identify whether there are fully-ambiguous traces in the language or not (cf. Lemma 2). We first need the following technical definition.

Definition 24. A cycle in G_{test3} is said to be F_i-indeterminate if the corresponding cycles in G_{d1} and G_{d2} are both F_i-indeterminate.

Such notion is helpful in providing a test to check whether the system is diagnosable under Protocol 3 or not. The following result introduces the test.

THEOREM 11 *A live and prefix-closed language L is diagnosable with respect to Protocol 3, the set of projections P_1, P_2, and the failure partition Π_f on Σ_f if and only if for all failure types F_i the following is true: G_d does not have F_i-indeterminate cycles* **and** *G_{test3} does not have F_i-indeterminate cycles.*

Proof: Sufficiency($\Leftarrow$). G_d and G_{test3} do not have F_i-indeterminate cycles for all failure types F_i. Consider a trace $st \in L(G)$ such that $s \in \Psi(\Sigma_{fi})$, and t is long enough, i.e., $\|t\| > n$, where n can be arbitrarily large. By the construction of G_{test3} and the assumptions above, s cannot lead to F_i-indeterminate cycles in both G_{d1} and G_{d2}. Therefore G_{d1} and G_{d2} do not simultaneously loop in F_i-indeterminate cycles, which implies that either G_{d1} or G_{d2} will diagnose the failure. Since s is arbitrary, L is diagnosable under Protocol 3.

Necessity($\Rightarrow$). Assume $L(G)$ is diagnosable under Protocol 3. Consider a trace $st \in L(G)$ such that $s \in \Psi(\Sigma_{fi})$, and t is long enough, i.e., $\|t\| > n$, where n can be arbitrarily large. By definition this implies that C is F_i-certain ($F_i \in C$ more precisely) in a finite number of steps along st. Therefore, the state of one of the diagnosers G_{d1} or G_{d2} is F_i-certain (in a finite number of steps) along st by the specification of Protocol 3. Hence, G_{test3} does not enter an F_i-indeterminate cycle. It is easy to verify that if the state of one of the diagnosers G_{d1} or G_{d2} is F_i-certain then so is the state of G_d. Therefore G_d enters an F_i-certain state in a finite number of steps along st, i.e., G_d does not have F_i-indeterminate cycles. Since s is arbitrary, then for all failure types F_i, G_d and G_{test3} do not have F_i-indeterminate cycles. ■

6.5. Discussion

We first note that the performance of Protocol 3 is inferior to that of Protocol 2 in the sense that there are more restrictions on the system structure for Protocol 3 to perform as well as the centralized diagnoser than there is for Protocol 2. This can be seen by the following example.

Example 17. Consider the system discussed in Example 12 and shown in Figure 8. In Example 14 we showed that with the FSM representation shown in Figure 8, Protocol 2 performs as well as the centralized diagnoser. G_{test3} for the system of Example 12 is shown in Figure 14. The cycle labeled A in the Figure is a F_1-indeterminate cycle: it can be verified that the cycles $(5F1, 8N)$ and $(4F1, 9N)$ are F_1-indeterminate cycles in G_{d1} and G_{d2}, respectively; hence by Theorem 11 the system is not diagnosable under Protocol 3.

Protocol 2 performs better than Protocol 3 because failure-ambiguous traces are also fully-ambiguous, however the reverse is not always true. However, the communication, processing, and memory requirements for Protocol 3 are less than those of Protocol 2. Indeed, the diagnostic information generated at the local sites, in case of Protocol 3, is a subset of the coordinator's diagnostic information in the case of Protocol 2. In addition, communication is significantly reduced in Protocol 3, and the decision rule does not involve any processing at the coordinator.

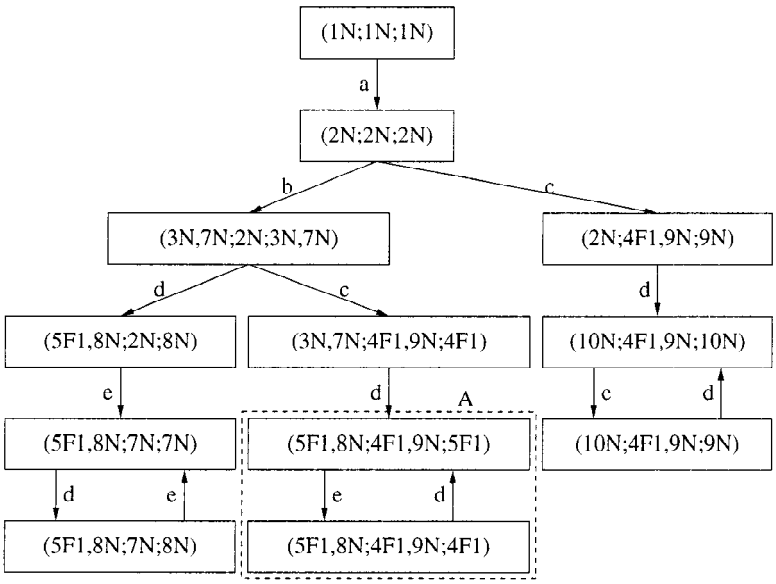

Figure 14. G_{test3} for Example 17.

As is the case for Protocol 2, the partitioning of observable events is crucial in deciding whether a language is diagnosable under Protocol 3 or not. Changing the partitions P_1 and P_2 may eliminate the presence of fully-ambiguous traces.

7. Discussion

In this section we discuss some fundamental issues related to the coordinated failure diagnosis protocols presented in this paper.

7.1. "Performance vs. Complexity" Tradeoff

As is the case with all coordinated decentralized architectures, the issue of "performance vs. complexity" should be addressed. By "performance vs. complexity" we understand a discussion of the qualitative properties of the protocols in terms of how well they perform and what their memory and processing power requirements are at the local sites and at the coordinator site. The presentation of Protocols 1–3 in Sections 4–6 was done in a manner that highlights this tradeoff. Protocol 1 performs as well as the centralized diagnoser irrespective of the system structure and the partitioning of observable events. Protocol 2 achieves the same task while constraining the system structure, and Protocol 3 adds additional constraints to those of Protocol 2 to achieve the diagnostic performance of the centralized diagnoser. Note here that the performance of Protocols 2 and 3 depend on the partitioning

Table 4. Comparison of the three protocols.

Protocol	Constraints on system structure	Partitioning of observable events
1	None	irrelevant
2	No failure-ambiguous traces	relevant
3	No fully-ambiguous traces	relevant

Table 5. Comparison of the three protocols (continued).

Protocol	Diagnostic information	Information communicated	Communication instances	Decision rule
1	Extended diagnoser state, extended unobservable reach	Extended diagnoser state, extended unobservable reach, status bit	upon each observable event occurrence	two intersections
2	Diagnoser state, unobservable reach	Diagnoser state, unobservable reach, status bit	upon the coordinator's request[3]	one intersection
3	Diagnoser state	Failure label	upon the diagnoser state being failure certain	no intersection

of observable events. Therefore, the diagnostic performance of the protocols improves from 3 to 2 to 1. However, the memory and processing requirements for implementing the protocols increases from 3 to 2 to 1: from a considerably low amount of processing and communication and a simple decision rule for Protocol 3 to more processing and communication, and a more involved decision rule for Protocol 2, to more processing and communication and an even more complicated decision rule for Protocol 1. Tables 4 and 5 summarize the above comparison.

7.2. Issues in Ordering Messages

Since the model we are using is untimed, one cannot talk about communication delay in numerical terms. Instead, the issue of order is of importance: since no time stamps are assumed to be available, we need to order the occurrence of events in the order of their execution by the system. Assumptions **A4** and **A5** address this issue, since they guarantee the arrival of messages in the order they are sent locally and globally. However, part of this assumption, namely the preservation of the global order, is admittedly strong. From network theory (Bertsekas and Gallager, 1992), we know that sender/receiver protocols that guarantee the correct and ordered reception of messages at the receiver end exist, and they apply to one network layer connection. Therefore, one can assume that the communication between one local site and the coordinator is correct and ordered. In the

case of multiple connections, which is of interest to us since we have two or more sites that communicate to the same receiver, protocols cannot always guarantee the correct and globally ordered reception of messages at the receiver. If Assumption **A5** is violated, the performance of Protocol 1 degrades drastically to the extent of possibly generating false positives. Before presenting an example that illustrates this scenario, we specify how the relaxation of Assumption **A5** may be approached.

Assume that the global order of reception of messages at the coordinator is not preserved and that there exists an upper bound, say **T** units, for the delay experienced by a message before it reaches the coordinator. By a unit delay we understand the maximum time elapsed between the execution of two consecutive events in the system. In this case, instead of having registers to hold the latest received messages, we have a buffer for each such register. The top of the buffer contains the earliest received message, or in other words the buffer is a first-in first-out queue. Once the new diagnostic coordinator information C is updated, a timer is set equal to **T** and the next update of C is done after the timer expires. However, since the global order of reception of messages is not preserved, the coordinator's information update is not as easy as explained in Section 4. The coordinator should find a way to order the occurrence of events before it updates its information and applies its decision rule. There are cases where this ordering can be achieved easily by checking the status bits: if both bits are equal to 1 this means that the event is common and the coordinator applies the corresponding decision action. However, the case of two consecutive events when the first is observed by one site and the second by the other site needs special attention. In such a case, the coordinator may apply the two possible information updates; if one of them results in an empty intersection, the other update is the correct one since the true system state necessarily belongs to both diagnoser states or to their unobservable reaches. If both updates result in non empty intersections not only this approach may fail but it may generate false positives as the following example demonstrates.

Example 18. Consider the system discussed in Example 1 and shown in Figure 2. Consider that the system is executing the trace $abc(de)^*$, the maximum delay is 4 units and Protocol 1 is used. Initially the timer is set to 4 units when the system begins executing its events. Therefore, after the timer expires, the coordinator may begin trying to order the occurrence of events. To do so the coordinator tests decision actions **DR4** and **DR1**. Decision action **DR1**, which is the correct action to take since event a occurred first, results in a candidate non empty coordinator diagnostic information C. Decision action **DR4**, which assumes that event b occurred first, results also in a candidate non empty C since there is a legal system behavior that begins with an event b, namely the trace $bac\sigma(de)^*$. Therefore the coordinator is not capable of ordering the occurrence of events at this stage. A possible choice would be to consider both possibilities and wait for another **T** units of time and try to figure out which is the correct order of occurrence of events. If so, the coordinator is faced with testing whether ab or ba is the correct sequence of occurrence of events by applying a similar test as the one we discussed earlier. This test will not solve the problem since both orders represent a legal behavior in the system after the occurrence of two events. Hence the coordinator is still unable to identify the sequence of occurrence of the events. After that the continuation of the sequences ab, $abc(de)^*$, or ba, $bac\sigma(de)^*$ have the same

projections with respect to P_1 and P_2 indefinitely, and by applying the suggested technique the coordinator would not be able to resolve the confusion about which trace was executed by the system. This confusion lasts indefinitely, and whatever messages the coordinator will save, it will not be able to resolve it. Furthermore, if the coordinator picks the trace $bac\sigma(de)^*$, then it may declare a false positive by asserting that a failure of type F_1 has occurred. Note here that since the coordinator does not have a copy of the system model then if the suggested algorithm were to work then it would have to be implemented at the coordinator site as a set of instructions (without reference to the rationale behind the algorithm).

The above example highlights a fundamental difficulty arising in untimed models of coordinated decentralized systems with partial observations, namely that of ordering: based on the information available from local sites, the coordinator is not always capable of determining the correct order of occurrence of events. A similar problem is also discussed in Wong and van Schuppen (1996).

The above example reveals a fundamental limitation of the untimed DES models and decentralized architectures used in this paper. To relax assumptions such as Assumption **A5**, we submit that timed discrete-event models have to be used.

7.3. Extension to m Sites

In our discussion, we have considered the generic case of two sites. The results obtained in this paper can be extended to the case of m sites in a straightforward manner. We will not state and prove these results, but we will give a logical explanation why they should stand.

In Protocol 3, the extension is quite obvious: in the case of m diagnosers, one of the m diagnosers should be able to identify the occurrence (the type) of any failure that occurs. In order to verify such a condition, we extend G_{test3} to include the synchronization of all the local diagnosers, in addition to the centralized one, and check for the existence of cycles where the corresponding cycles of all local diagnosers are indeterminate.

The case of Protocol 2 is more involved. The communication rules are the analogues of **CR1** and **CR2** for all local sites, and the decision rule can be extended in the following way: in case the event is common to all, intersect all the states of the diagnosers, otherwise intersect the states of all diagnosers who saw the event with the unobservable reaches of all diagnosers who did not see the event. A mechanism to identify who saw the last event and who did not should be implemented at the coordinator site. Using such rule, the test to check diagnosability is to identify the existence of ambiguous cycles in a machine that represents the extension of G_{test2} to m sites. Such a test will provide the correct result since the intersection operator is associative.

Finally, in the case of Protocol 1, we adopt the same extension of the communication rules as for Protocol 2. The decision rule can be extended in the same way, the only difference being that $\cap_e^i$ is used instead of the regular intersection. Moreover, after applying the intersection to the states and unobservable reaches, the operator $\cap_c$ is applied to identify the admissible behavior. Since $\cap_e^i$ is associative by definition, the decision rule is reconstructing the centralized diagnoser state at the coordinator site; hence, there exists a test, namely the

test on the centralized diagnoser, to check whether Protocol 1 diagnoses the system or not. Note here that in such a case, a mechanism should be implemented at the coordinator site to take care of who saw what and when.

7.4. General Thoughts on the Approach

Two salient features of the decentralized protocols presented in this paper are: (1) the diagnostic algorithms employed at the local sites are based on centralized diagnosis procedures, that is, diagnostic information at each local site is generated by solving a centralized diagnosis problem at the site as in Sampath et al. (1995) or Sampath (1993); (2) the objective is to determine realizations of the architecture of Section 3.1 that perform as well as the centralized diagnostic scheme.

The use of diagnosers or extended diagnosers at the local sites is guided by the powerful results of Sampath et al. (1995) and Sampath (1993) on the centralized diagnosis problem. Even though the use of centralized diagnosis procedures provides a reasonable strategy for generating diagnostic information at the local sites, it is far from clear that such procedures always present the best alternative. For example, it may be possible to achieve the same performance as Protocol 1 if at the local sites we use diagnostic algorithms other than extended diagnosers. Such algorithms could take into account the fact that diagnostic information is generated at more than one sites, they could utilize the knowledge that is common (Aumann, 1976; Washburn and Teneketzis, 1984) to all local sites, and could lead to protocols that perform as well as a centralized diagnoser with less requirements on communication, data processing and memory than Protocol 1. The discovery of such protocols is a very challenging open problem with far reaching implications.

As pointed out in Section 3.3, the performance of the centralized architecture proposed in Sampath et al. (1995) provides an upper bound on the performance achievable (i.e., the failure events diagnosable) by any realization of the coordinated decentralized architecture of Section 3.1. Achieving the performance of the centralized diagnostic scheme proposed in Sampath et al. (1995) with a coordinated decentralized architecture requires a certain amount of resources for data storage and data processing both at the local sites and the coordinator, as well as a certain amount of bandwidth for data communication, specified by Protocol 1. When the amount of resources for data storage, processing and communication is limited, the challenge is to determine the best performance achievable under the resource constraint. The presence of such a constraint gives rise to problems that are significantly more difficult than the ones studied in this paper, because currently there are no tight upper bounds on the diagnostic performance, achievable under a resource constraint, to guide the design of decentralized coordinated protocols. The determination of such tight bounds is an important fundamental problem with significant practical implications.

8. Conclusion

In this paper, we have extended the theory of diagnosability of systems in the framework of formal languages (Sampath et al., 1995; Sampath, 1995) to a class of coordinated decentral-

ized systems. We presented three coordinated decentralized protocols that are capable under certain assumptions of diagnosing all failure types diagnosed by the centralized diagnoser. We identified necessary and sufficient conditions for diagnosability under the proposed protocols. The key feature of the presented protocols is that they highlight the "performance vs. complexity" tradeoff that appears in coordinated decentralized architectures. The online diagnostic process is carried through the diagnosers (extended diagnosers in the case of Protocol 1) implemented at the local sites, i.e., the scheme is indeed implemented in a decentralized fashion.

Our analysis has been based on a set of assumptions, some of which, namely the liveness of the language and the nonexistence of cycles of unobservable events, can be relaxed easily as discussed in Sampath (1995) and Sampath et al. (1998). However, the assumptions on ordering are critical and reveal some fundamental limitations of the untimed DES models and decentralized architectures used in this paper.

Acknowledgements

We would like to thank the reviewers for their comments and suggestions which helped improve the presentation and readability of the paper. This research was supported in part by NSF grant ECS-9509975 and by the Department of Defense Research & Engineering (DDR&E) Multidisciplinary University Research Initiative (MURI) on "Low Energy Electronics Design for Mobile Platforms" and managed by the Army Research Office (ARO) under grant ARO DAAH04-96-1-0377.

Notes

1. This theorem is presented as an unproved claim in Sampath (1993).
2. We commit a slight abuse of notation by using the Kleene closure * in the expression of a trace.
3. As explained in Section 5.5 and not as presented in Section 5.2.2.

References

Aumann, R. J. 1976. Agreeing to disagree. *The Annals of Statistics* 4(6): 1236–1239.

Bavishi, S., and Chong, E. 1994. Automated fault diagnosis using a discrete event systems framework. *Proc. 9th IEEE International Symposium on Intelligent Control*, pp. 213–218.

Bertsekas, D., and Gallager, R. 1992. *Data Networks.* Englewood Cliffs, NJ: Prentice Hall.

Boubour, R., Jard, C., Aghasaryan, A., Fabre, E., and Benveniste, A. 1997. Petri net approach to fault detection and diagnosis in distributed systems: application to telecommunication networks, motivations, and modelling. *Proc. 36th IEEE Conf. on Decision and Control*. San Diego, CA.

Cassandras, C. G., and Lafortune, S. 1998. Discrete event systems: The state of the art and some recent trends. *Applied and Computational Control, Signals and Circuits* (Datta, B, ed.). Birkhäuser.

Castanon, D. A., and Teneketzis, D. 1985. Distributed estimation algorithms for nonlinear systems. *IEEE Trans. Automat. Contr.* 30(5): 418–425.

Davis, R., and Hamscher, W. 1992. Model based reasoning: Troubleshooting. *Readings in Model Based Diagnosis* (Hamscher, W., Console, L., and Kleer, J., eds.). Morgan Kaufman, pp. 3–24

Debouk, R. Failure Diagnosis of Discrete Event Systems with Distributed Information. PhD Thesis, Electrical Engineering and Computer Science Department, The University of Michigan. In preparation.

Frank, P. 1990. Fault diagnosis in dynamic systems using analytical and knowledge based redundancy. *Automatica* 26: 459–474.

Holloway, L., and Chand, S. 1994.Time templates for discrete event fault monitoring in manufacturing systems. *Proc. 1994 American Control Conference*, pp. 701–706.

Hopcroft, J., and Ullman, J. 1979. *Introduction to Automata Theory, Languages, and Computation*. Reading, MA: Addison Wesley.

Lapp, S., and Powers, G. 1977. Computer-aided synthesis of fault trees. *IEEE Trans. Reliability Engineering* 26(1): 2–13.

Lin, F. 1994. Diagnosability of discrete event systems and its application. *Discrete Event Dynamic Systems: Theory and Applications* 4(2): 197–212.

Pouliezos, A. D., and Stavrakakis, G. S. 1994. *Real Time Fault Monitoring of Industrial Processes.* Boston, MA: Kluwer Academic Publishers.

Ramadge, P. J., and Wonham, W. M. 1989. The control of discrete-event systems. *Proc. IEEE* 77(1): 81–98.

Sampath, M. 1993. The extended diagnoser. Unpublished memorandum.

Sampath, M. 1995. A discrete event systems approach to failure diagnosis. PhD thesis, The University of Michigan.

Sampath, M., Lafortune, S., and Teneketzis, D. 1998. Active diagnosis of discrete-event systems. *IEEE Trans. Automat. Contr.* 43(7).

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. 1995. Diagnosability of discrete-event systems. *IEEE Trans. Automat. Contr.* 40(9): 1555–1575.

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. 1996. Failure diagnosis using discrete-event models. *IEEE Trans. Contr. Syst. Tech.* 4(2): 105–124.

Scherer, W. T., and White, C. C. 1987. A survey of expert systems for equipment maintenance and diagnostics. In *Fault Detection and Reliability: Knowledge Based & Other Approaches* (Singh, M. G., Hindi, K. S., Schmidt, G., and Tzafestas, S., eds.). Pergamon Press.

Speyer, J. L. 1979. Computation and transmission requirements for a decentralized linear-quadratic-gaussian control problem. *IEEE Trans. Automat. Contr.* 24: 266–269.

Washburn, R. B., and Teneketzis, D. 1984. Asymptotic agreement among communicating decision makers. *Stochastics* 13(1–2): 103–129.

Willsky, A. 1976. A survey of design methods for failure detection in dynamic systems. *Automatica* 12: 601–611.

Willsky, A. S., Bello, M., Castanon, D. A., Levy, B. C., and Verghese, G. 1982. Combining and updating of local estimates and regional maps along sets of one-dimensional tracks. *IEEE Trans. Automat. Contr.* 27(4): 799–813.

Wong, K., and van Schuppen, J. H. 1996. Decentralized supervisory control of discrete event systems with communications. *Proc. of WODES 1996, International Workshop on Discrete Event Systems*. IEE, London, England, pp. 284–289.