On the Effect of Communication Delays in Failure Diagnosis of Decentralized Discrete Event Systems

Rami Debouk, Stéphane Lafortune and Demosthenis Teneketzis
Department of EECS, The University of Michigan
1301 Beal Avenue, Ann Arbor, MI 48109–2122, USA
{ridebouk,stephane,teneketzis}@eecs.umich.edu; www.eecs.umich.edu/umdes

Abstract

We study the effect of communication delays on the performance of a coordinated decentralized architecture for failure diagnosis of untimed discrete event systems. The architecture consists of local sites communicating with a coordinator that is responsible for diagnosing the failures occurring in the system. A protocol that realizes the architecture is defined by the diagnostic information generated at the local sites, the communication rules used by the local sites, and the decision rule used by the coordinator to infer the occurrence of failures. Our prior work [6] has addressed the performance of a set of protocols under the assumption that messages sent from various local sites to the coordinator are received in the order in which they are sent globally. In this work we relax the abovementioned assumption. We modify the coordinator's decision rule for one of the protocols analyzed in [6] to account for the reception of out of order messages at the coordinator's site. We discover conditions on the system structure under which the modified protocol performs as well as the centralized diagnostic scheme proposed in [11].

1 Introduction

Failure detection and isolation is an important task in the automatic control of large complex systems. Diagnosing systems not only improves their performance and productivity, but also protects life and property. For these reasons, approaches for failure diagnosis have been extensively studied in the literature [10]. Almost all of the failure diagnosis approaches in the literature have been developed for systems where the information used for fault diagnosis is centralized. The majority of technologically complex systems (computer and communication networks, manufacturing systems, 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. Most approaches to failure diagnosis suggested in the literature do not apply directly to informationally decentralized systems, hence the need to develop diagnostic methodologies for these systems. This fact is also recognized in [1], [4], [7], [8], [9], and [12]. The philosophy of the approach we are suggesting is totally different from that of [7] and [8]. Although similarities exist between our approach and that of [1], [4], [9], and [12], the modeling and/or informational assumptions of [1], [4], [9], and [12] and our work are clearly different. The reader is referred to [5] for a brief description of the approaches of [1], [4], [7], [8], [9], and [12], and a comparison of theses approaches to the work presented in this paper.

In this paper, we restrict attention to a coordinated decentralized architecture for failure diagnosis with two local sites communicating with a coordinator. This architecture is depicted in Figure 1. The top block

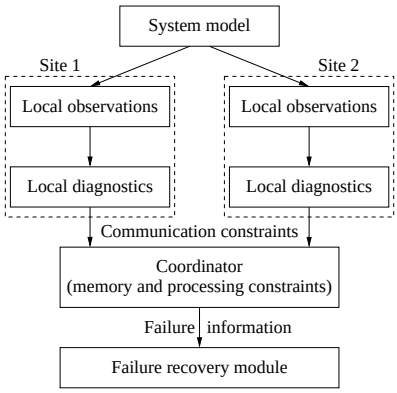

Figure 1: Coordinated decentralized architecture

represents the complete system model. The system is modeled by a regular language which accounts for both normal and failed modes of operation of the system. The language is defined over an alphabet which is the union of the disjoint sets of observable events and unobservable events. The set of failures to be diagnosed is a subset of the set of unobservable events. Each site is composed of two modules: an observation module and a diagnostic module. Site i, $i \in \{1, 2\}$, locally observes the system, and based on its own observations

generates its own diagnostic information. Both sites communicate some form of their diagnostic information (specified in Section 4) to the coordinator. 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.

In [6] we investigated the diagnosability properties of the above architecture under a set of assumptions. A major assumption was that the messages communicated to the coordinator are received in the order they are sent globally. However, this assumption may be too restrictive for decentralized systems. Local sites are using, in general, different links to route their messages to the coordinator, and consequently messages sent from different sites may be received out of order at the coordinator due to communication or propagation delays. The objective of this paper is to study the effect of these delays and consequently the effect of out of order reception of messages, on the performance of coordinated decentralized diagnostic protocols. For that matter, we concentrate on one of the diagnostic protocols presented in [6], namely Protocol 2, and relax the assumption that global ordering of the messages at the coordinator's site is maintained. We call the resulting diagnostic scheme Protocol 2D. We discover conditions sufficient to guarantee that Protocol 2D performs as well as the centralized diagnostic scheme proposed in [11]. These conditions are exactly the same ones that ensure that Protocol 2 performs as well as the diagnostic scheme of [11]. Consequently, relaxing the assumption on global ordering of messages at the coordinator's site does not result in performance degradation. However, Protocol 2D requires more memory storage at the coordinator's site than Protocol 2.

We note here that, due to space limitations, the technical results of this paper are presented without proofs. The reader is referred to [5] for a complete account of these results.

2 Preliminaries

This section introduces some preliminary definitions, assumptions, and notations needed for the development of the technical results of this paper. The reader is referred to [11] for more details and a formal description of some "loosely" defined terms and notations.

2.1 The system model

The system to be diagnosed is modeled as a deterministic finite state machine (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 $L(G)$ generated by G. $L(G)$ is a subset of Σ^*, where Σ^* denotes the Kleene closure of the set Σ. In this paper we will use the language $L(G)$, or simply L, and the system interchangeably.

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 projection $P : \Sigma \to \Sigma_o$ is defined in the usual manner [3]. Let $\Sigma_f \subseteq \Sigma$ denote the set of failure events which are to be diagnosed. We assume, without loss of generality, that $\Sigma_f \subseteq \Sigma_{uo}$. 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 sets corresponding to different failure types

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

Let Π_f denote this partition. Hereafter, when we write that a failure of type F_i has occurred, we will mean that some event of the set Σ_{fi} has occurred.

The diagnoser is a deterministic FSM built from the system model G. This machine is at the core of the diagnostic methodology of [11]. It is used to analyze off-line the diagnosability properties of G and to perform diagnostics when it observes on-line the behavior of the system. We refer the reader to [11] for a presentation of the construction of the diagnoser.

2.2 Diagnosis in the coordinated decentralized architecture

In decentralized systems, the global system information is distributed at several sites. The "agents" (decision makers, diagnosers, etc.) at different sites may communicate and exchange information in real time, or just report some or all of their information to a center that, in general, possesses limited knowledge of the system. In this paper we consider the coordinated decentralized architecture depicted 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. Each site is composed of two modules: an observation module and a diagnostic module. 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}; 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 observations and generates its own diagnostic information. Both sites communi-

cate 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.

Following the above discussion, in order 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. 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. We have the following three definitions from [6].

Definition 2.1 *Within the context of the coordinated decentralized architecture described in Section 1 and depicted in Figure 1, a* **protocol** *is defined by the diagnostic rules used at the local sites, the rules employed by the local sites to communicate their diagnostic information to the coordinator, and the coordinator's decision rule.*

Definition 2.2 *The coordinator's diagnostic information C is said to be* $\mathbf{F_i}$**-certain** *if based on C, the coordinator is certain that a failure of type F_i has occurred.*

Note that the diagnostic information C is protocol dependent.

Definition 2.3 *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*

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

where $s \in \Psi(\Sigma_{fi})$ denotes the fact that the last event of s is a failure event of type F_i, L/s denotes the post-language of L after s, and $\| t \|$ denotes the length of trace t.

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

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

1:$L(G)$ is live[1].
2: G has no cycles of unobservable events with respect to either Σ_{o1} or Σ_{o2}.
3: $L(G)$ is not diagnosable (in the centralized setup of [11]) with respect to P_i and Π_f on Σ_f, $i = 1, 2$.
4: 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.
5: The sets of observable events at each site are common knowledge [15] to all sites.
6 The two sites are allowed to report to the coordinator only a processed version of their raw data.
7: The coordinator does not have a model of the system, that is, it does not know the dynamics of the system.

We note that in this work, in contrast to [6], there is no assumption that global order of messages is maintained. That is, the messages transmitted by different (distinct) local sites are not necessarily received at the coordinator's site in the order in which they are sent. In fact, and as mentioned earlier, *this paper studies the effect of out of order message reception at the coordinator's site on the performance of decentralized diagnostic protocols.*

3 Addressing communication delay issues in the coordinated decentralized architecture

In coordinated decentralized architectures, like the one in Figure 1, the coordinator may receive messages out of order. Ordering of messages from one site to the coordinator may be easily achieved with a transport layer protocol, for instance TCP, or a data link control layer protocol such as Go Back n or Selective Repeat [2]. However, global order, that is, ordering of messages sent by different local sites to the coordinator is not necessarily maintained. To achieve global ordering of messages we may/can use one of two possible approaches/ mechanisms. (i) We can introduce clocks at the local sites and time-stamp the messages sent by the local sites to the coordinator. In this situation we need to make sure that clocks are synchronized; we can achieve clock synchronization by the use of Global Positioning Systems (GPS) (see [13]). Such a mechanism/approach is not always feasible. For example, synchronizing clocks may be too constraining in low-energy mobile communication networks [14]. (ii) We can use untimed discrete event models and design algorithms that order the messages arriving at the coordinator's site.

In this paper we will use the second approach. Such an approach is needed for instance in telecommunication networks [9] where time information is not available.

[1] A language L is said to be live if for all $s \in L$, there exists $\sigma \in \Sigma$ such that $s\sigma \in L$.

Under this approach, to generate a diagnostic decision we store all incoming messages up until the time where all possible orders in which these messages are sent by various local sites can be sorted out. Once these orders are sorted out, the coordinator's decision rule is applied to all possible orders. The same procedure is repeated every time incoming messages arrive at the coordinator's site. To highlight the approach we assume that messages sent by local sites are received at the coordinator at most "one-step out of order". To illustrate this assumption consider the following scenario. Assume the system executes the sequence of events ab. Suppose that event a (resp. b) is observed by local site 1 (resp. local site 2). Denote by x (resp. y) the message generated by the occurrence of event a (resp. b). If the order of reception of messages at the coordinator's site is yx, then x is said to be received "1-step out of order". Hence, the assumption says that a message received at the coordinator could have been received in the correct order it is sent before (resp. after) the message that was (resp. will be) received before (resp. after) it.

4 Protocol 2D: a coordinated decentralized protocol

In this section, we modify one of the three protocols considered in [6] to account for communication delays. We will refer to the modified protocol as **Protocol 2D** (where 2D stands for the fact that is a modification of Protocol **2** of [6] that allows for communication **D**elays). We determine conditions under which the protocol is capable of diagnosing the same types of failures as the ones diagnosed using the centralized diagnostic scheme of [11].

As is the case for any protocol that realizes the coordinated decentralized architecture, we need to define the diagnostic information generated at the local sites, the communication rules used by the local sites, and the decision rule used by the coordinator to infer the occurrence of failures. This is done in the following three subsections.

4.1 Diagnostic information at local sites

As is the case with Protocol 2, diagnosers are implemented at local sites. Consequently, 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 formally defined in [6]. By definition the diagnoser state only represents those states that are reached following an observable event; the unobservable reach of a diagnoser state at site j appends to the diagnoser state the states that are reached through unobservable events (to site j) following that observable event up to an event observable by the other site.

4.2 Communication rules

The communication rules as defined for Protocol 2 are used for Protocol 2D. Communication rule [**CRi**], $i = 1, 2$, says that 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 $\sigma \notin \Sigma_{oj}$.

4.3 Decision rule

The coordinator's decision rule is composed of three steps: (1) storing all incoming messages at the coordinator site, and sorting out all possible orders in which these messages were generated by the local sites; (2) applying the information update rule, as defined in Section 5.1.3 in [6], to each and every possible order, and retaining all orders that result in a non-empty update (intersection); (3) comparing the failure properties of all surviving updates from step (2), and declaring the occurrence of a failure when all these updates are certain of the occurrence of the failure. These steps are discussed in the following three sub-sections. Note that the following analysis assumes that there are no events commonly observed by sites 1 and 2. The case where there are events that are commonly observed at both sites is briefly discussed in Section 4.5.

4.3.1 Sorting out possible orders: All communication messages received at the coordinator's site are stored until the instance where all possible orders in which these messages are generated by the local sites can be figured out. Determining the instance where all possible orders can be sorted out depends on the messages received, namely which site observed an event and subsequently sent the message. In general, one can continue sorting out (uncovering) all possible orders after waiting for the arrival of at most three new messages at the coordinator's site. This is a direct consequence of the "one-step out of order" assumption and it is explained below.

Table 1 depicts the arrival of three new messages at the coordinator's site. A message with subscript i, $i \in \{1, 2\}$, indicates it has been sent by site i. The left

Messages	Possible orders	Sorted out orders
$x_1 y_2 z_2$	$x_1 y_2 z_2$ or $y_2 x_1 z_2$	$x_1 y_2$ or $y_2 x_1$
$x_2 y_1 z_1$	$x_2 y_1 z_1$ or $y_1 x_2 z_1$	$x_2 y_1$ or $y_1 x_2$
$x_1 y_2 z_1$	$x_1 y_2 z_1$ or $y_2 x_1 z_1$ or $x_1 z_1 y_2$	$x_1 y_2$ or $y_2 x_1$ or $x_1 z_1 y_2$
$x_2 y_1 z_2$	$x_2 y_1 z_2$ or $y_1 x_2 z_2$ or $x_2 z_2 y_1$	$x_2 y_1$ or $y_1 x_2$ or $x_2 z_2 y_1$

Table 1: Sorting out possible orders at the coordinator site

column in Table 1 describes the order in which messages

are received at the coordinator's site (left is earliest), the middle column describes all possible orders that may have resulted in the reception of messages by the coordinator, and the right column describes the orders sorted out. The second column of the first row in Table 1 means the following: either the reception order is the same as the one in which the messages are sent, or y_2 could have been sent prior to x_1 (due to the "one-step out of order" communication delay). In both cases z_2 is sent after y_2 due to preservation of local order. The third column of the first row in Table 1 means the following: the sorted out (uncovered) orders are x_1y_2 or y_2x_1 and z_2 needs to be stored until new messages are received to figure out the order in which it was generated. The second row is the symmetric to the first and the last two rows are interpreted in a similar way. At reset, the "order sorting" procedure is to wait for three messages and afterwards uncover all possible orders of the first two messages while keeping the third message for later consideration after new messages arrive[2].

4.3.2 Application of the update rule: Before defining the information update rule of Protocol 2D, we recall the structure of the coordinator from [6]. For every possible order that is sorted out 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 of protocol 2 of [6] is specified in Table 2.

At any instant of time when all possible orders can be sorted out the coordinator considers these orders and applies to each order the information update rule as defined in Table 2. For every possible order that survives the update rule, the coordinator keeps the last update along with the corresponding states and unobservable reaches and status bits. This is possible by definition of the update rule that only requires information from the last update to generate the new one (cf. Table 2). If a specific order of messages results in an empty intersection, the coordinator rejects that as an impossible order. The information is stored by the coordinator until the next instant of time when new possible orders can be extracted (as a continuation of the already existing ones). Then the same procedure as above is applied to these new orders and the new updates replace the old ones that are discarded.

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

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

Having defined the information update rule, we now provide more details on how the "order sorting" procedure introduced in Section 4.3.1 and the information update rule are implemented. Initially, the coordinator waits until receiving three messages (as depicted in Table 1) before sorting out the possible new orders. At that instant orders including only two messages are sorted out, and the information update rule is applied to these orders. The registers for each sorted out order are updated to hold the newest diagnostic update along with the corresponding states and unobservable reaches and status bits. Also stored for each order is the third message (cf. Table 1). When two new messages are received at the coordinator site, the coordinator sorts out the new possible orders (out of the third message that was previously stored for each existing sorted out order and the two new messages that have been received) as a continuation of a previously uncovered order. To every new sorted out order, the coordinator applies the information update rule. By successively applying the procedure just described, the coordinator sorts out the possible orders after receiving two new messages, except for reset steps where three new messages are needed. In fact, a reset step is either the first time the coordinator attempts sorting out messages (discussed above) or when a previous sorting attempt results in an uncovered order that contains all the received messages (cf. the last uncovered orders in rows three and four in Table 1 and the discussion in footnote 2). In general, at any instant of time when the coordinator has received $2n+1$ messages, the orders of at least $2n$ messages are uncovered by the arguments discussed above.

4.3.3 Diagnosing failures: If all the information updates that result from applying the procedure

[2] Note here that in the case of possible orders $x_1z_1y_2$ and $x_2z_2y_1$ in rows three and four, respectively, the order in which the three messages are sent is uncovered since by the "one-step out of order" assumption messages y_2 and y_1, received at the coordinator's site prior to messages z_1 and z_2, should have been sent directly after these messages.

described in Sections 4.3.1 and 4.3.2 are certain that a failure of type F_i has occurred, then the coordinator declares that F_i has occurred and broadcasts this information to the failure recovery module. Otherwise, a diagnostic decision is postponed.

4.4 Diagnostic properties of Protocol 2D

The decision rule discussed in Section 4.3 has the following two features: (1) The memory requirements at the coordinator site may increase considerably because the coordinator has to store all the above-mentioned information updates and the corresponding diagnoser states and unobservable reaches. However, the amount of memory required remains finite since the models used are finite-state automata (a full account of this issue is available in [5]). (2) The coordinator identifies a failure only when that failure is identified by all surviving information updates. Therefore, we expect that, in general, the performance of Protocol 2D will not be the same as that of Protocol 2 where global order is preserved. Interestingly enough, we prove next that the absence of *failure-ambiguous* traces, is a condition sufficient to ensure that Protocol 2D performs as well the centralized diagnostic scheme of [11]. We note that Protocol 2 in [6] performs as well as the centralized diagnostic scheme of [11] under the same condition. Thus, relaxing the assumption on global ordering of messages at the coordinator's site does not result in any performance degradation of the protocol from that viewpoint. To proceed with the analysis of Protocol 2D we first present the notion of failure-ambiguous traces as defined in [6].

Definition 4.1 *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*[3]*, 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''.*

The next theorem summarizes the main result concerning the diagnostic performance of Protocol 2D.

[3]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 the length of s is greater than n and s possesses the given property.

Theorem 4.1 *Under the assumption that messages are received at the coordinator at most "one-step out of order", Protocol 2D eventually identifies all failure types that are detected by the centralized diagnostic scheme of [11] if there are no failure-ambiguous traces (with respect to all failure types).*

We conjecture that the result of Theorem 4.1 still holds even when messages are received at the coordinator at most "n-steps out of order". We expect that the delays associated with diagnostic decisions and the memory requirements at the coordinator's site will increase with n.

4.5 Procedure using common events

The procedure presented in Section 4.3 assumed that all messages communicated to the coordinator regard only events that are observed by either site 1 or site 2, but not both. In fact, if there are events that are commonly observed by both sites then these events can be used as a synchronization mechanism. Since local order is preserved, the coordinator knows how to order the messages that are due to commonly observed events. As a result of the information update rule at the coordinator's site, messages generated by commonly observed events need to contain only the states of the local diagnosers (and not their unobservable reaches). Therefore, under the assumption that commonly observed events are executed frequently along all traces, the protocol can be modified as follows: local sites use the diagnosers to generate their diagnostic information; they communicate their diagnosers' states to the coordinator only after the occurrence of commonly observed events; the decision rule of the coordinator is to apply the information update rule as specified in Table 1. Denote by **Protocol 2D-C** (where **C** stands for commonly observed events) the above specified protocol. Then, we have the following result.

Theorem 4.2 *If there no failure-ambiguous traces (with respect to all failure types), Protocol 2D-C eventually identifies the same failures as the centralized diagnostic scheme of [11].*

4.6 Polling procedure

Another approach is to advise the coordinator to poll the sites requesting each site to communicate its current state and unobservable reach plus the status bit specifying whether the event that led to the current state was observed by the other site or not. Denote this protocol by **Protocol 2D-P** (where **P** stands for polling). Under the assumption that the polling messages are received at the two sites before the system executes a new observable event (in Σ_o), we have the following result.

Theorem 4.3 *Under the assumption that the polling messages are received at the two sites before the system executes a new event, Protocol 2D-P eventually identifies the same failures as the centralized diagnostic scheme of [11] if there are no failure-ambiguous traces (with respect to all failure types).*

Protocol 2D-P saves on communication and processing power and memory storage (a comparable memory storage to the case of Protocol 2 is needed). It avoids as well the delaying of diagnostic decisions when the frequency of occurrence of commonly observed events is low. Also, the "one-step out of order" assumption is not needed as long as communication delays are bounded.

5 Discussion

In this paper, we have extended the theory of diagnosability of decentralized discrete event systems. We have presented a coordinated decentralized protocol (Protocol 2D; and two variations of it, Protocols 2D-C and 2D-P) that is capable, under certain conditions, of diagnosing all failure types diagnosed by the centralized diagnostic scheme of [11]. The on-line diagnostic process is carried through the diagnosers implemented at the local sites, i.e., the scheme is indeed implemented in a decentralized fashion.

The key features of Protocol 2D are: (1) it achieves the same performance as the centralized diagnostic scheme of [11] when there are no failure-ambiguous traces. That is, the absence of global ordering of the messages received at the coordinator's site does not affect diagnosing failures (compared to Protocol 2 of [6]). (2) The delay of its diagnostic decision is higher than the delay of the centralized diagnostic scheme of [11]. (3) The memory required at the coordinator's site to implement the protocol is larger than that required in Protocol 2 of [6]. The first feature of Protocol 2D is a bit surprising, whereas its last two features are not unexpected.

Acknowledgements This research was supported in part by NSF grants ECS-9509975 and ECS-0080406, 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.

References

[1] P. Baroni, G. Lamperti, P. Pogliano, and M. Zanella. Diagnosis of large active systems. *Artificial Intelligence*, 110:135–183, 1999.

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

[3] C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems.* Kluwer Academic Publishers, Boston, MA, 1999.

[4] S. Deb, A. Mathur, P. Willett, and K.R. Pattipati. De-centralized real-time monitoring and diagnosis. In *Proc. IEEE Conf. on Systems, Man and Cybernetics*, pages 2998–3003, October 1998.

[5] R. Debouk. *Failure Diagnosis of Decentralized Discrete Event Systems.* PhD thesis, Electrical Engineering and Computer Science Department, The University of Michigan, 2000. Available at `http://www.eecs.umich.edu/umdes`.

[6] R. Debouk, S. Lafortune, and D. Teneketzis. Coordinated decentralized protocols for failure diagnosis of discrete-event systems. *Journal of Discrete Event Dynamical Systems: Theory and Applications*, 10:33–86, 2000.

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

[8] S. Mohindra and P.A. Clark. A distributed fault diagnosis method based on digraph models: Steady-state analysis. *Computers and Chemical Engineering*, 17(2):193–209, 1993.

[9] Y. Pencolé. Decentralized diagnoser approach: Application to telecommunication networks. In *Proc. of DX'2000, Eleventh International Workshop on Principles of Diagnosis*, pages 185–192, June 2000.

[10] A.D. Pouliezos and G.S. Stavrakakis. *Real time fault monitoring of industrial processes.* Kluwer Academic Publishers, Boston, MA, 1994.

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

[12] R. Sengupta. Diagnosis and communication in distributed systems. In *Proc. of WODES 1998, International Workshop on Discrete Event Systems*, pages 144–151. Published by IEE, London, England, August 1998.

[13] U. Schmid, Guest Editor. Special issue on global time in large scale distributed real time systems. *Real Time Systems*, 12(I, II, and III):1–351, Jan, Mar, and May 1997.

[14] W. Stark, Project Director. Low energy electronics design for mobile platforms. Project Report of ARO-MURI for the period 9/96 to 7/99, Electrical Engineering and Computer Science Department, The University of Michigan, 1999.

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