The Diagnosis of Failures via the Combination of Distributed Observations

Kurt R. Rohloff

Abstract— **This paper discusses a computational problem related to the distributed diagnosis of system behavior in the discrete-event systems framework. A new framework for distributed diagnosis is introduced where local observers periodically transmit their observations to a centralized coordinator that diagnoses failures in the system. A condition called combined observation diagnosability is shown for the centralized coordinator to be able to diagnose system failures. A construction is shown to test a system for combined observation diagnosability.**

I. Introduction

An important problem in many modern systems is to diagnose and respond to the occurrence of system failures. Unfortunately, failure occurrences in a system commonly cannot be observed directly. Therefore, failure diagnosis would have to be performed indirectly through the observation of other behaviors in the system. That is, based on observed behavior, a diagnoser should be able to eventually determine without ambiguity if a failure event has occurred. Related to this scenario is the problem of distributed diagnosis. Commonly a system may be too large or complex to be observed in a centralized manner. Therefore, distributed observations would have to be made of the system behavior and then the task of the diagnoser would be to use the combined information of the distributed observations to determine if a failure had occurred. This paper investigates the problem of distributed diagnosis in the framework of discrete-event systems modeled as automata.

Some of the early innovative work in discrete-event systems diagnosis in [4, 10–12]. The diagnosis problem continues to be an active research area, but of particular relevance to the discussions in this paper is the problem of distributed diagnosis which is also discussed in [3, 6]. The authors in [3, 6] present results related to decentralized diagnosis problems with automata system models where the local observers, in a sense, encode their local observations as a local state.

However, there are fundamental limitations to the state-based encoding of local observations for use in conjunction with diagnosis, even for finite state systems. The problem is considered in [5] where two observers make local observations of the behavior of a finite state system and the observers communicate encodings of their local observations to a centralized coordinator for the diagnosis the system's state. It is shown in [5] that even for a simple system there exists no finite encoding of locally observed system behavior such that the centralized coordinator can perform diagnosis as well as if the local observers had communicated all of their local observations. This shows that even for finite state systems as discussed in [3, 6] there is a loss of information when the local observations are encoded as the state of a finite state machine. Therefore, a distributed diagnosis system could be more accurate if the full local observations are saved and used for diagnosis directly by a centralized coordinator.

To avoid the limitations due to information loss resulting from observation encodings as discussed above, this paper proposes a different distributed diagnosis framework where instead of communicating finite state encodings of the locally observed behavior, the observers store the locally observed behavior, and these observations are periodically transmitted in batch form to a centralized coordinator. An example of such a system might be a constellation of satellites that are at times unable to communicate with a ground station due to the paths of their orbits, but when the satellites can communicate with the ground station, the communication cost is relatively inexpensive. Because this paper assumes a logical untimed system model, the term "periodically" is used loosely to denote behaviors which are guaranteed to occur infinitely often over all possible infinite behavior paths of a system rather than behaviors which occur at regular time intervals.

The framework of periodic communication can be thought of as a compromise between the seminal centralized diagnosis systems of [11] and the standard distributed frameworks as discussed in [3, 6]. That is, due to the periodic communications of local observations in this framework the centralized coordinator can perform diagnosis with eventual access to all available local information when traditional centralized diagnosis is not feasible. It should be noted that even though this distributed diagnosis system has full access to the local observations, the distributed system is still not equivalent to a centralized system [1]. Therefore, a new property called combined observation diagnosability is presented below to describe systems that can be diagnosed using this new framework. Also, a construction is presented to decide if a system is combined observation diagnosable that is similar to the state reachability construction methods discussed in [9] for decentralized control.

The next section presents the notation and other preliminaries related to the distributed diagnosis problem. Section III discusses the new distributed diagnosis framework and

This research was supported by the NSF grant CCR 00-85917 ITR.

K. Rohloff is with the Coordinated Science Laboratory at The University of Illinois, 1308 West Main St., Urbana, IL 61801, USA. krohloff@control.csl.uiuc.edu

diagnosability definition. Section IV presents a method for testing if a system is combined observation diagnosable. Section V closes the paper with a discussion of the results contained herein. Due to reasons of brevity, the proofs of results in this paper are demonstrated in the companion journal version of this paper, [8], in a generalized setting.

II. Notation and Preliminaries

The failure diagnosis notation from [3, 11] and the ω-language notation of [7] is now introduced. The systems to be diagnosed are modeled as automata, where for the system $G = (X, \Sigma, x_0, \delta, X_m)$, X is a set of states, Σ is an event set, x_0 is the initial state and $\delta : X \times \Sigma \to X$ is the possibly partial transition function. The set of marked states $X_m \subseteq X$ is sometimes empty in which case X_m is not listed in the components of G. The definition of $\delta(\cdot, \cdot)$ can be extended in the usual manner to be defined over strings. It is assumed that the systems discussed in this paper are deterministic.

Following the commonly used definitions of discrete event systems, G has a generated language $\mathcal{L}(G)$ such that

$$\mathcal{L}(G) = \{s \in \Sigma^* | \delta(x_0, s)!\} \tag{1}$$

where $\delta(x_0, s)!$ is true if and only if $\delta(x_0, s)$ is defined.

For a string of events $s = \sigma_1\sigma_2\sigma_3\cdots$, possibly infinite, let $s(i)$ denote $s = \sigma_1\sigma_2\sigma_3\cdots\sigma_i$. The set Σ^ω is all infinitely long traces of events in Σ. Formally,

$$\Sigma^\omega = \{\sigma_1\sigma_2\sigma_3\cdots | \forall i \in \mathbb{N}, \sigma_i \in \Sigma\}. \tag{2}$$

For the automaton G, the generated ω-language of G, denoted $\mathcal{L}^\omega(G)$, is the set of all infinitely long traces of behavior which are possible in G. Formally,

$$\mathcal{L}^\omega(G) = \{s \in \Sigma^\omega | \forall i \in \mathbb{N}, s(i) \in \mathcal{L}(G)\}. \tag{3}$$

Similarly, for an automaton G, the marked ω-language of G, denoted $\mathcal{L}^\omega_m(G)$, is the set of all traces of behavior in $\mathcal{L}^\omega(G)$ where a marked state is visited infinitely often. Formally,

$$\mathcal{L}^\omega_m(G) = \{s \in \mathcal{L}^\omega(G) | \forall i \in \mathbb{N}, \exists j \in \mathbb{N} : s(i+j) \in \mathcal{L}_m(G)\}. \tag{4}$$

For a language $L \subseteq \Sigma^*$ and a string $s \in \overline{L}$, the set L/s is the set of all finite strings s' such that ss' is in L. Formally,

$$L/s = \{s' \in \Sigma^* | ss' \in L\}.$$

A similar definition exists if $L \subseteq \Sigma^\omega$.

For a string $s \in \Sigma^*$, $\|s\|$ is the length of s. The notation is also used that for a set of events $\Sigma_a \subseteq \Sigma$,

$$\Psi(\Sigma_a) = \{s \in \Sigma^*\Sigma_a\Sigma^*\}.$$

That is, $\Psi(\Sigma_a)$ is the set of all finite strings that contain an event in Σ_a.

Also, for a string $s \in \Sigma^*$, the set $\Phi(s) \subseteq \Sigma$ are the events that comprise s. Formally,

$$\Phi(s) = \{\sigma \in \Sigma | s \notin (\Sigma \setminus \{\sigma\})^*\}. \tag{5}$$

For the modeling of the failure events, the set of system events Σ has a subset of failure events $\Sigma^f \subseteq \Sigma$ of which all are assumed to be unobservable. It is assumed that for the set of failure events Σ^f there is a partition of failure classes $\Pi^f = \{\Sigma^{f1}, \cdots, \Sigma^{fn}\}$ such that

$$\Sigma^f = \cup_{i \in \{1,\ldots,n\}} \Sigma^{fi}.$$

The partitioning of failure events captures the behavior that all of the failures within a certain class are equivalent with respect to failure responses. Therefore, after a failure occurs, knowledge of the exact failure does not matter as long as it is known that a failure of a certain type has occurred.

Using the standard partial observation notation from [2], a subset of the system events $\Sigma_o \subseteq \Sigma$ could be directly observable by a centralized diagnoser, and $\Sigma_{uo} = \Sigma \setminus \Sigma_o$ represents the set of unobservable events. There is a projection operation $P : \Sigma^* \to \Sigma_o^*$ along with a corresponding inverse projection operation $P^{-1} : \Sigma_o^* \to 2^{\Sigma^*}$.

With the basic notation presented, diagnosability for standard centralized diagnosis systems is shown.

Definition 1: [11] A prefix-closed and live language L is said to be *diagnosable* with respect to a projection P and a partition Π^f if the following holds

$$\left(\forall \Sigma^{fi} \in \Pi^f\right)\left(\exists n_i \in \mathbb{N}\right)\left(\forall s \in \Psi(\Sigma^{fi})\right)\left(\forall s' \in L/s\right) \left(\|s'\| \geq n_i \Rightarrow D\right) \tag{6}$$

where the diagnosability condition D is

$$\left[\forall t \in P^{-1}\left(P(ss')\right) \cap L\right]\left(\Sigma^{fi} \cap \Phi(t) \neq \emptyset\right) \tag{7}$$

Intuitively, L is diagnosable if for every failure class Σ^{fi} and a string s containing a failure in Σ^{fi}, then there is always a string s' of length less than n_i such that if $P(ss')$ is observed by a centralized diagnoser, then the diagnoser knows for sure that a failure in the class Σ^{fi} occurred.

As discussed in [11], the liveness assumption is made for simplicity and is not necessary. This definition is generalized in [3] for situations where the system might have some arbitrary diagnosis protocol.

III. Combined Observation Diagnosis

As was discussed above, it is not always feasible to use centralized diagnosis protocols on a system G which is too large or complex for a diagnoser to make observations in a centralized manner. In this scenario, a distributed diagnosis system has decentralized observers O_1 and O_2 that make distributed observations which are communicated periodically to a centralized coordinator. Based on the information from the observers, the coordinator performs the diagnosis operation.

To formalize the problem, let the distributed observers O_1 and O_2 observe occurrences of the events in Σ_{o1} and Σ_{o2} respectively. The set of observable events Σ_o is the union of Σ_{o1} and Σ_{o2}. As system behavior progresses in the system, the observers record the local observations $P_1(s)$ and $P_2(s)$ of the behavior s that occurred in the system. Note that $P_1(\cdot)$ and $P_2(\cdot)$ are the observation projections

for O_1 and O_2 defined in the usual manner along with the corresponding inverse operations $P_1^{-1}(\cdot)$ and $P_2^{-1}(\cdot)$.

Let $\Sigma_t \subseteq \Sigma_{o1} \cap \Sigma_{o2}$ be the set of events which trigger the observers to transmit their stored observations to the coordinator. That is, for $\sigma_t \in \Sigma_t$, if O_1 observers $\alpha\beta\sigma_t\gamma\kappa\sigma_t$, then on the occurrence of the first σ_t, $\alpha\beta\sigma_t$ is transmitted to the coordinator, and on the occurrence of the second σ_t, $\gamma\kappa\sigma_t$ is transmitted to the coordinator.

Suppose a string $s \in \mathcal{L}(G) \cap \Sigma^*\Sigma_t$ has occurred in the system. Due to s, O_1 observes $P_1(s) \in \Sigma_{o1}^*\Sigma_t$ and O_2 to observes $P_2(s) \in \Sigma_{o2}^*\Sigma_t$. Note that the last event in s is a trigger event, so due to the framework in which the observers communicate to the coordinator on the occurrence of Σ_t events, then the full observations $P_1(s)$ and $P_2(s)$ are known by the coordinator after s occurs. Using the communicated observations $P_1(s)$ and $P_2(s)$ and the following theorem, the coordinator can then perform failure diagnosis operations.

Theorem 1: Suppose a system G, sets of observable events Σ_{o1} and Σ_{o2}, a set of trigger events Σ_t and a string $s \in \mathcal{L}(G) \cap \Sigma^*\Sigma_t$ is given. A string t generates the same set of communications $P_1(s)$ and $P_2(s)$ from the observers to the coordinator as s if and only if

$$t \in \left[\begin{array}{l} \left[P_1^{-1}(P_1(s))(\Sigma_{o1} \setminus \Sigma_t)^*\right] \cap \\ \left[P_2^{-1}(P_2(s))(\Sigma_{o1} \setminus \Sigma_t)^*\right] \cap \mathcal{L}(G) \end{array} \right]. \quad (8)$$

Because $\left(P_1^{-1}(P_1(s))(\Sigma_{o1} \setminus \Sigma_t)^*\right) \cap \left(P_2^{-1}(P_2(s))(\Sigma_{o1} \setminus \Sigma_t)^*\right) \cap \mathcal{L}(G)$ represents all strings in $\mathcal{L}(G)$ that could have generated the communications $P_1(s)$ and $P_2(s)$ to the coordinator from O_1 and O_2 respectively, then $\left(P_1^{-1}(P_1(s))(\Sigma_{o1} \setminus \Sigma_t)^*\right) \cap \left(P_2^{-1}(P_2(s))(\Sigma_{o1} \setminus \Sigma_t)^*\right) \cap \mathcal{L}(G) \subseteq \Sigma^*\{f\}\Sigma^*$ if and only if the failure event f must have occurred in the system given the observations $P_1(s)$ and $P_2(s)$. Similarly, $\left(P_1^{-1}(P_1(s))(\Sigma_{o1} \setminus \Sigma_t)^*\right) \cap \left(P_2^{-1}(P_2(s))(\Sigma_{o1} \setminus \Sigma_t)^*\right) \cap \mathcal{L}(G) \subseteq (\Sigma \setminus \{f\})^*$ if and only if f could not have occurred in the system given the communicated observations $P_1(s)$ and $P_2(s)$. Note that when given $P_1(s)$ and $P_2(s)$, then $\left(P_1^{-1}(P_1(s))(\Sigma_{o1} \setminus \Sigma_t)^*\right) \cap \left(P_2^{-1}(P_2(s))(\Sigma_{o1} \setminus \Sigma_t)^*\right) \cap \mathcal{L}(G)$ can be computed in polynomial time with respect to the sizes of $P_1(s)$ and $P_2(s)$ using standard methods.

It is generally assumed for simplicity in the rest of this paper that for any string of behavior that has occurred in a system, an event in Σ_t will always eventually occur. This assumption is made to ensure that for all strings of behavior in the system, the observers will eventually transmit their observations to the centralized coordinator. This effectively bounds the maximum size of any string that an observer needs to store and then transmit to the coordinator. This further ensures that the observers will not need infinite memory.

It is also assumed for simplicity that the system automata used in this paper are live as discussed in [11]. The following simplifying notations are also used:

$$\Sigma_{1\cap2} = \Sigma_{o1} \cap \Sigma_{o2} \quad (9)$$

$$\Sigma_{1\setminus2} = \Sigma_{o1} \setminus \Sigma_{o2} \quad (10)$$

$$\Sigma_{2\setminus1} = \Sigma_{o2} \setminus \Sigma_{o1}. \quad (11)$$

The set $\Sigma_{1\cap2}$ represents the events that can be observed by both observers, while events in $\Sigma_{1\setminus2}$ can be observed by observer 1 but not observer 2 and events in $\Sigma_{2\setminus1}$ can be observed by observer 2 but not observer 1.

For the distributed diagnosis protocol just presented, a corresponding version of diagnosability called combined observation diagnosability is now defined.

Definition 2: A prefix-closed and live language L is said to be *combined observation diagnosable* with respect to a projection P and a partition Π^f if the following holds

$$\begin{array}{r} \left(\forall \Sigma^{fi} \in \Pi^f\right)\left(\exists n_i \in \mathbb{N}\right)\left(\forall s \in \Psi(\Sigma^{fi})\right) \\ \left(\forall s' \in (L/s) \cap (\Sigma^*\Sigma_t)\right)\left(\|s'\| \geq n_i \Rightarrow D\right) \end{array} \quad (12)$$

where the diagnosability condition D is

$$\begin{array}{r} \left[\forall t \in P_1^{-1}\left(P_1(ss')\right) \cap P_2^{-1}\left(P_2(ss')\right) \cap L\right] \\ \left(\Sigma^{fi} \cap \Phi(t) \neq \emptyset\right) \end{array} \quad (13)$$

Intuitively, L is combined observation diagnosable if for every failure class Σ^{fi} and a string s containing a failure in Σ^{fi}, then there is always a string s' of length less than n_i that ends with an event in Σ_t such that it can be known for sure with information of the local observations $P_1(ss')$ and $P_2(ss')$ that a failure in the class Σ^{fi} occurred. The condition that $s' \in (L/s) \cap (\Sigma^*\Sigma_t)$ is used because after the occurrence of Σ_t events, the observers transmit their observed strings to the centralized coordinator. By definition, the property of combined observation diagnosability is necessary and sufficient for there to exist a set of distributed observers that periodically transmit their observations to a centralized coordinator that can always eventually diagnose faults.

Note that $P_1^{-1}\left(P_1(ss')\right) \cap P_2^{-1}\left(P_2(ss')\right) \cap L$ is not always equivalent to $P^{-1}\left(P(ss')\right) \cap L$. This implies that there might be languages which are diagnosable but not combined observation diagnosable. This in fact is true, but not shown here due to reasons of brevity.

IV. The Testing of Combined Observation Diagnosability

A method is know shown to test combined observation diagnosability. For simplicity it is assumed that $|\Sigma^f| = 1$ so that there is exactly one failure event f. The results and methods presented in this section are easily generalized for the more general case of $|\Sigma^f| \neq 1$.

It is also assumed without loss of generality that the state space X of G can be partitioned into two subsets X^n and X^f such that $X^n \cup X^f = X$ and $X^n \cap X^f = \emptyset$ where

$$\begin{array}{l} X^n = \\ \quad \{x \in X | \exists s \in \mathcal{L}(G) \cap \left(\Sigma \setminus \Sigma^f\right)^* : \delta(x_0, s) = x\} \end{array} \quad (14)$$

and

$$\begin{array}{l} X^f = \\ \quad \{x \in X | \exists s \in \mathcal{L}(G) \cap \left(\Sigma^*\Sigma^f\Sigma^*\right) : \delta(x_0, s) = x\}. \end{array} \quad (15)$$

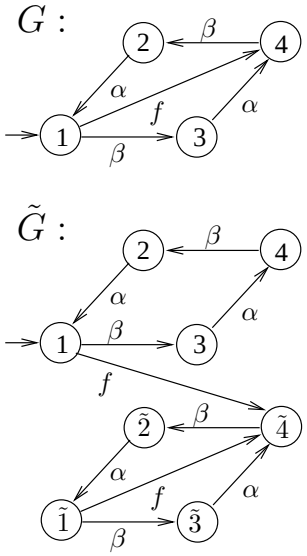

Fig. 1. Constructing $\tilde{G}$ from G.

The set of states X^n are all states the system could be in if no failure events have occurred while X^f are all states the system could be in if at least one failure event has occurred. The set X^n is also called the *normal* states while X^f is called the *failure* states.

If the partition of system states X into normal and failure states as described above does not exist for the given automaton G, then another automaton $\tilde{G} = (\tilde{X}, \Sigma, x_0, \tilde{\delta})$ can be easily constructed from G such that the normal and failure state partitions exist and $\mathcal{L}(G) = \mathcal{L}(\tilde{G})$. Furthermore, the size of the state space of $\tilde{G}$ is at most double the size of the state space of G. To construct $\tilde{G}$, let $\tilde{X}'$ be a copy of X such that for every state $x \in X$ there is a corresponding state $\tilde{x} \in \tilde{X}'$. The state space of $\tilde{G}$ is the union $X \cup \tilde{X}'$. For the state transition function, suppose $x \in X$ and $\sigma \in \Sigma$. If $\delta(x, \sigma) = y$, then define $\tilde{\delta}(\tilde{x}, \sigma) = \tilde{y}$. If $\sigma \in \Sigma^f$, then define $\tilde{\delta}(x, \sigma) = \tilde{y}$, and if $\sigma \notin \Sigma^f$, then define $\tilde{\delta}(x, \sigma) = y$. Therefore, in $\tilde{G}$, the set of normal states is X and the set of failure states is $\tilde{X}'$. Also note that in general $\tilde{G}$ may not be trim in general.

An example of the method for constructing $\tilde{G}$ from G is now shown in Example 1.

Example 1: Consider the system G in Figure 1 where f is a failure event. The system $\tilde{G}$ constructed from G as outlined above can also be seen in Figure 1. Note that $\mathcal{L}(G) = \mathcal{L}(\tilde{G})$ and in $\tilde{G}$, the set of states X is $\{1, 2, 3, 4\}$ and the set of states $\tilde{X}'$ is $\{\tilde{1}, \tilde{2}, \tilde{3}, \tilde{4}\}$.

In order to test combined observation diagnosability, for the set of events $\Sigma \setminus \{f\}$, let there be another set of events Σ' such that $\Sigma \cap \Sigma' = \emptyset$ and there is a one-to-one mapping $\Lambda : \Sigma \to \Sigma'$ such that for every event $\sigma \in \Sigma \setminus \{f\}$ there is a corresponding event $\Lambda(\sigma) \in \Sigma'$ and $\Lambda(f)$ is undefined. The mapping $\Lambda(\cdot)$ can be extended in the usual manner to be defined over strings of events, sets of events and languages. There is also a corresponding inverse mapping $\Lambda^{-1} : \Sigma' \to \Sigma \setminus \{f\}$. The event $\Lambda(\sigma)$ is sometimes written as σ' when it can be done without ambiguity. Similar notation is also used for sets of events and strings such that $\Lambda(\Sigma_a) = \Sigma'_a$ for $\Sigma_a \subseteq \Sigma$ and $\Lambda(t) = t'$ for $t \in \Sigma^* \cup \Sigma^\omega$. Also define the projections $P_\Sigma : \Sigma \cup \Sigma' \to \Sigma$ and $P_{\Sigma'} : \Sigma \cup \Sigma' \to \Sigma'$ in the usual manner.

To test the combined observation diagnosability of the language generated by $\mathcal{G}$, an automaton $\mathcal{D} = (X^\mathcal{D}, \Sigma^\mathcal{D}, x_0^\mathcal{D}, \delta^\mathcal{D}, X_m^\mathcal{D})$ is constructed such that $\mathcal{L}_m^\omega(\mathcal{D})$ is empty if and only if $\mathcal{L}(G)$ is combined observation diagnosable with respect to the local projections $P_1(\cdot)$ and $P_2(\cdot)$ and the failure event f. Intuitively, the automaton construction $\mathcal{D}$ tracks to see if there is a pair of infinite strings $(s_1, s_2) \in (\Sigma^*\{f\}\Sigma^\omega) \times (\Sigma \setminus \{f\})^\omega$ such that $\{s_1, s_2\} \subseteq \mathcal{L}^\omega(G)$, and for any prefix $s_1' \in \overline{s_1} \cap (\Sigma^*\{f\}\Sigma^*\Sigma^t)$, there is a corresponding prefix $s_2' \in \overline{s_2} \cap \left((\Sigma \setminus \{f\})^* \Sigma^t\right)$ such that $P_1(s_1') = P_1(s_2')$ and $P_2(s_1') = P_2(s_2')$. If such a pair (s_1, s_2) exists, then there is a finite string that is a prefix of s_1 that contains a failure f and ends with a Σ_t event that cannot be distinguished from a prefix of s_2 ending with a Σ_t event and no failure event.

The components of $\mathcal{D}$ are defined as follows:

$$X^\mathcal{D} = (X^n \cup X^f) \times X^n \times R_1 \times R_2 \times R_{1\cap 2} \quad (16)$$

$$\Sigma^\mathcal{D} = \Sigma \cup \Sigma' \quad (17)$$

$$x_0^\mathcal{D} = (x_0, x_0, \epsilon, \epsilon, \epsilon) \quad (18)$$

$$X_m^\mathcal{D} = X^f \times X^n \times \{\epsilon\} \times \{\epsilon\} \times \{\epsilon\} \quad (19)$$

where $R_1 = \Sigma_{1\setminus 2}^*, R_2 = \Sigma_{2\setminus 1}^*, R_{1\cap 2} = \Sigma_{1\cap 2} \cup \{\epsilon\}$.

The notation is sometimes used such that $\vec{x} \overset{\gamma}{\mapsto}_\mathcal{D} \vec{y}$ represents that according to the transition rules of $\mathcal{D}$ there is a transition from state $\vec{x}$ to state $\vec{y}$ labelled by event γ. Therefore, $\vec{y} \in \delta^\mathcal{D}(\vec{x}, \gamma)$ if an only if $\vec{x} \overset{\gamma}{\mapsto}_\mathcal{D} \vec{y}$. The state transition representations are also extended in the usual manner to be defined over strings of transitions labelled by strings of events.

Note that in $\mathcal{D}$ there are two classes of events, Σ and Σ' that are used to form strings of behaviors $\vec{s} \in \Sigma^{\mathcal{D}*}$. The string of events $\vec{s}$ is used to track the possible pairs of behaviors (s_1, s_2) in G as discussed above where $s_1 = P_\Sigma(\vec{s})$, $s_2 = \Lambda^{-1}(P_{\Sigma'}(\vec{s}))$, $P_1(s_1) = P_1(s_2)$, $P_2(s_1) = P_2(s_2)$ and $f \notin \Phi(s_2)$. The transition structure of $\mathcal{D}$ is constructed such that $\vec{x}_1 \overset{\vec{s}}{\mapsto}_\mathcal{D} \vec{x}_2$ where $\vec{x}_1 = (x_1^1, x_1^2, \epsilon, \epsilon, \epsilon)$ and $\vec{x}_2 = (x_2^1, x_2^2, \epsilon, \epsilon, \epsilon)$ if and only if for $s_1 = P_\Sigma(\vec{s})$ and $s_2 = \Lambda^{-1}(P_{\Sigma'}(\vec{s}))$ in G, $\delta(x_1^1, s_1) = x_2^1$, $\delta(x_1^2, s_2) = x_2^2$, $P_1(s_1) = P_1(s_2)$ and $P_2(s_1) = P_2(s_2)$. The state transition function $\delta^\mathcal{D} : X^\mathcal{D} \times \Sigma^\mathcal{D} \to 2^{X^\mathcal{D}}$ is defined as follows.

$$\delta^\mathcal{D}((x_1, x_2, \epsilon, \epsilon, \epsilon), \sigma) = \begin{cases} (\delta(x_1, \sigma), x_2, \epsilon, \epsilon, \epsilon) & \text{if } \sigma \in \Sigma_{uo} \\ (x_1, \delta(x_2, \Lambda^{-1}(\sigma)), \epsilon, \epsilon, \epsilon) & \text{if } \sigma \in \Sigma'_{uo} \\ (\delta(x_1, \sigma), x_2, \epsilon, \epsilon, \sigma) & \text{if } \sigma \in \Sigma_{1\cap 2} \\ (\delta(x_1, \sigma), x_2, \sigma, \epsilon, \epsilon) & \text{if } \sigma \in \Sigma_{1\setminus 2} \\ (\delta(x_1, \sigma), x_2, \epsilon, \sigma, \epsilon) & \text{if } \sigma \in \Sigma_{2\setminus 1} \end{cases}$$

$$\delta^{\mathcal{D}}((x_1,x_2,\sigma_1 s_1,\epsilon,\epsilon),\sigma) =$$
$$\begin{cases} (\delta(x_1,\sigma),x_2,\sigma_1 s_1,\epsilon,\epsilon) & \text{if } \sigma\in\Sigma_{uo}\\ (x_1,\delta(x_2,\Lambda^{-1}(\sigma)),\sigma_1 s_1,\epsilon,\epsilon) & \text{if } \sigma\in\Sigma'_{uo}\\ (\delta(x_1,\sigma),x_2,\sigma_1 s_1,\epsilon,\sigma) & \text{if } \sigma\in\Sigma_{1\cap 2}\\ (\delta(x_1,\sigma),x_2,\sigma_1 s_1\sigma,\epsilon,\epsilon) & \text{if } \sigma\in\Sigma_{1\setminus 2}\\ (\delta(x_1,\sigma),x_2,\sigma_1 s_1,\sigma,\epsilon) & \text{if } \sigma\in\Sigma_{2\setminus 1}\\ (x_1,\delta(x_2,\Lambda^{-1}(\sigma)),s_1,\epsilon,\epsilon) & \text{if } \sigma=\Lambda(\sigma_1)\end{cases}$$

$$\delta^{\mathcal{D}}((x_1,x_2,\epsilon,\sigma_2 s_2,\epsilon),\sigma) =$$
$$\begin{cases} (\delta(x_1,\sigma),x_2,\epsilon,\sigma_2 s_2,\epsilon) & \text{if } \sigma\in\Sigma_{uo}\\ (x_1,\delta(x_2,\Lambda^{-1}(\sigma)),\epsilon,\sigma_2 s_2,\epsilon) & \text{if } \sigma\in\Sigma'_{uo}\\ (\delta(x_1,\sigma),x_2,\epsilon,\sigma_2 s_2,\sigma) & \text{if } \sigma\in\Sigma_{1\cap 2}\\ (\delta(x_1,\sigma),x_2,\sigma,\sigma_2 s_2,\epsilon) & \text{if } \sigma\in\Sigma_{1\setminus 2}\\ (\delta(x_1,\sigma),x_2,\epsilon,\sigma_2 s_2\sigma,\epsilon) & \text{if } \sigma\in\Sigma_{2\setminus 1}\\ (x_1,\delta(x_2,\Lambda^{-1}(\sigma)),\epsilon,s_2,\epsilon) & \text{if } \sigma=\Lambda(\sigma_2)\end{cases}$$

$$\delta^{\mathcal{D}}((x_1,x_2,\sigma_1 s_1,\sigma_2 s_2,\epsilon),\sigma) =$$
$$\begin{cases} (\delta(x_1,\sigma),x_2,\sigma_1 s_1,\sigma_2 s_2,\epsilon) & \text{if } \sigma\in\Sigma_{uo}\\ (x_1,\delta(x_2,\Lambda^{-1}(\sigma)),\sigma_1 s_1,\sigma_2 s_2,\epsilon) & \text{if } \sigma\in\Sigma'_{uo}\\ (\delta(x_1,\sigma),x_2,\sigma_1 s_1,\sigma_2 s_2,\sigma) & \text{if } \sigma\in\Sigma_{1\cap 2}\\ (\delta(x_1,\sigma),x_2,\sigma_1 s_1\sigma,\sigma_2 s_2,\epsilon) & \text{if } \sigma\in\Sigma_{1\setminus 2}\\ (\delta(x_1,\sigma),x_2,\sigma_1 s_1,\sigma_2 s_2\sigma,\epsilon) & \text{if } \sigma\in\Sigma_{2\setminus 1}\\ (x_1,\delta(x_2,\Lambda^{-1}(\sigma)),s_1,\sigma_2 s_2,\epsilon) & \text{if } \sigma=\Lambda(\sigma_1)\\ (x_1,\delta(x_2,\Lambda^{-1}(\sigma)),\sigma_1 s_1,s_2,\epsilon) & \text{if } \sigma=\Lambda(\sigma_2)\end{cases}$$

$$\delta^{\mathcal{D}}((x_1,x_2,\epsilon,\epsilon,\sigma_{1\cap 2}),\sigma) =$$
$$\begin{cases} (\delta(x_1,\sigma),x_2,\epsilon,\epsilon,\sigma_{1\cap 2}) & \text{if } \sigma\in\Sigma_{uo}\\ (x_1,\delta(x_2,\Lambda^{-1}(\sigma)),\epsilon,\epsilon,\sigma_{1\cap 2}) & \text{if } \sigma\in\Sigma'_{uo}\\ (\delta(x_1,\sigma),\delta(x_2,\Lambda^{-1}(\sigma)),\epsilon,\epsilon,\epsilon) & \text{if } \sigma=\Lambda(\sigma_{1\cap 2})\end{cases}$$

$$\delta^{\mathcal{D}}((x_1,x_2,\sigma_1 s_1,\epsilon,\sigma_{1\cap 2}),\sigma) =$$
$$\begin{cases} (\delta(x_1,\sigma),x_2,\sigma_1 s_1,\epsilon,\sigma_{1\cap 2}) & \text{if } \sigma\in\Sigma_{uo}\\ (x_1,\delta(x_2,\Lambda^{-1}(\sigma)),\sigma_1 s_1,\epsilon,\sigma_{1\cap 2}) & \text{if } \sigma\in\Sigma'_{uo}\\ (x_1,\delta(x_2,\Lambda^{-1}(\sigma)),s_1,\epsilon,\sigma_{1\cap 2}) & \text{if } \sigma=\Lambda(\sigma_1)\end{cases}$$

$$\delta^{\mathcal{D}}((x_1,x_2,\epsilon,\sigma_2 s_2,\sigma_{1\cap 2}),\sigma) =$$
$$\begin{cases} (\delta(x_1,\sigma),x_2,\epsilon,\sigma_2 s_2,\sigma_{1\cap 2}) & \text{if } \sigma\in\Sigma_{uo}\\ (x_1,\delta(x_2,\Lambda^{-1}(\sigma)),\epsilon,\sigma_2 s_2,\sigma_{1\cap 2}) & \text{if } \sigma\in\Sigma'_{uo}\\ (x_1,\delta(x_2,\Lambda^{-1}(\sigma)),\epsilon,s_2,\sigma_{1\cap 2}) & \text{if } \sigma=\Lambda(\sigma_2)\end{cases}$$

$$\delta^{\mathcal{D}}((x_1,x_2,\sigma_1 s_1,\sigma_2 s_2,\sigma_{1\cap 2}),\sigma) =$$
$$\begin{cases} (\delta(x_1,\sigma),x_2,\sigma_1 s_1,\sigma_2 s_2,\sigma_{1\cap 2}) & \text{if } \sigma\in\Sigma_{uo}\\ (x_1,\delta(x_2,\Lambda^{-1}(\sigma)),\sigma_1 s_1,\sigma_2 s_2,\sigma_{1\cap 2}) & \text{if } \sigma\in\Sigma'_{uo}\\ (x_1,\delta(x_2,\Lambda^{-1}(\sigma)),s_1,\sigma_2 s_2,\sigma_{1\cap 2}) & \text{if } \sigma=\Lambda(\sigma_1)\\ (x_1,\delta(x_2,\Lambda^{-1}(\sigma)),\sigma_1 s_1,s_2,\sigma_{1\cap 2}) & \text{if } \sigma=\Lambda(\sigma_2)\end{cases}$$

Suppose that $\vec{x}=(x^1,x^2,r_1,r_2,r_{1\cap 2})$ is a state in $\mathcal{D}$ where $x^1,x^2\in X$, $r_1\in R_1$, $r_2\in R_2$, $r_{1\cap 2}\in R_{1\cap 2}$ and there is a string $\vec{s}$ such that $x_0^{\mathcal{D}}\overset{\vec{s}}{\mapsto}_{\mathcal{D}}\vec{x}$. In order to the track a pair of defined behaviors (s_1,s_2) in G where $s_1=P_\Sigma(\vec{s})$ and $s_2=\Lambda^{-1}(P_{\Sigma'}(\vec{s}))$, the component x^1 tracks the behavior due to s_1 in G and the component x^2 tracks the behavior due to s_2 in G. Hence, x^1 is the state of G resulting from the string of events $s_1=P_\Sigma(\vec{s})$ and the x^1 component in $\vec{x}$ only updates on the occurrence of events in Σ. Similarly, x^2 is the state of G resulting from the string of events $s_2=\Lambda^{-1}(P_{\Sigma'}(\vec{s}))$ and the x^2 component in $\vec{x}$ only updates on the occurrence of events in Σ'. The r_1 component of the state $\vec{x}$ in $\mathcal{D}$ is used to track the proper ordering of $\Sigma_{1\setminus 2}$ events in $\vec{s}$, the r_2 component of the state $\vec{x}$ in $\mathcal{D}$ is used to track the proper ordering of $\Sigma_{2\setminus 1}$ events in $\vec{s}$ and the $r_{1\cap 2}$ component of the state $\vec{x}$ in $\mathcal{D}$ is used to track the proper ordering of $\Sigma_{1\cap 2}$ events in $\vec{s}$ so that if $\vec{x}$ is reachable in $\mathcal{D}$ due to $\vec{s}$, then $P_1(s_1)=P_1(s_2 r_1 r_{1\cap 2})$ and $P_2(s_1)=P_2(s_2 r_2 r_{1\cap 2})$.

The r_1, r_2 and $r_{1\cap 2}$ state components operate in a similar manner as a queue such that if at $\vec{x}=(x^1,x^2,r_1,r_2,r_{1\cap 2})$ and an event $\sigma\in\Sigma_{1\setminus 2}$ occurs, then σ is appended to the end of r_1. If $\vec{x}=(x^1,x^2,r_1,r_2,r_{1\cap 2})$, then an event $\sigma'\in\Sigma'_{1\setminus 2}$ can only occur if $\Lambda^{-1}(\sigma')$ is at the front of the r_1 queue. Once $\sigma\in\Sigma'_{1\setminus 2}$ does occur, then the $\Lambda^{-1}(\sigma')$ is removed from the front of the r_1 queue. Taken as a whole, these restrictions ensure that in an infinitely long string of behaviors in $\mathcal{D}$, the relative order of $\Sigma'_{1\setminus 2}$ events in $\mathcal{D}$ is the same relative order of the corresponding $\Sigma_{1\setminus 2}$ events in $\mathcal{D}$. The queue for r_2 operates in a similar manner with respect to events in $\Sigma_{2\setminus 1}$.

Note that in the operation of $\mathcal{D}$, if an event $\sigma\in\Sigma_{1\cap 2}$ occurs, then no further $\Sigma_{1\setminus 2}$ events can occur until $\Lambda(\sigma)$ occurs and $\Lambda(\sigma)$ cannot occur until the r_1 queue is cleared. As an example, suppose a string $r_1\sigma_{1\cap 2}$ occurs in $\mathcal{D}$ where $r_1\sigma_{1\cap 2}\in\Sigma^*_{1\setminus 2}\Sigma_{1\cap 2}$. In order to ensure that for some future behavior, $P_1(P_\Sigma(\vec{s}))=P_1(\Lambda^{-1}(P_{\Sigma'}(\vec{s})))$ where $r_1\sigma_{1\cap 2}=P_1(P_\Sigma(\vec{s}))$, then it needs to be guaranteed that the occurrences of Σ_{o1} and Σ'_{o1} events are restricted such that $r_1\sigma_{1\cap 2}=P_1(\Lambda^{-1}(P_{\Sigma'}(\vec{s})))$. Therefore, the next string of Σ'_{o1} events to occur after $r_1\sigma_{1\cap 2}$ is necessarily $\Lambda(r_1\sigma_{1\cap 2})$ and the above restrictions on $\Sigma_{1\cap 2}$ and $\Sigma_{1\setminus 2}$ ensure this. Consequently, after a $\Sigma'_{1\cap 2}$ event occurs, $r_1=\epsilon$ and $r_{1\cap 2}=\epsilon$, and $P_1(P_\Sigma(\vec{s}))=P_1(\Lambda^{-1}(P_{\Sigma'}(\vec{s})))$. Similarly, if an event $\sigma\in\Sigma_{1\cap 2}$ occurs, then no further $\Sigma_{2\setminus 1}$ events can occur until $\Lambda(\sigma)$ occurs and $\Lambda(\sigma)$ cannot occur until the r_2 queue is cleared.

Consequently, after a $\Sigma'_{1\cap 2}$ event occurs and the system is in some state $\vec{x}$ such that $\vec{x}=(x^1,x^2,\epsilon,\epsilon,\epsilon)$ and $x_0^{\mathcal{D}}\overset{\vec{s}}{\mapsto}_{\mathcal{D}}\vec{x}$, then $P_2(P_\Sigma(\vec{s}))=P_2(\Lambda^{-1}(P_{\Sigma'}(\vec{s})))$. Therefore, if $\vec{x}_1\overset{\vec{s}}{\mapsto}_{\mathcal{D}}\vec{x}_2$ where $\vec{x}_1=(x_1^1,x_1^2,\epsilon,\epsilon,\epsilon)$ and $\vec{x}_2=(x_2^1,x_2^2,\epsilon,\epsilon,\epsilon)$, then $P_1(P_\Sigma(\vec{s}))=P_1(\Lambda^{-1}(P_{\Sigma'}(\vec{s})))$ and $P_2(P_\Sigma(\vec{s}))=P_2(\Lambda^{-1}(P_{\Sigma'}(\vec{s})))$. This is formalized below in the proof of Theorem 3

Theorem 2: Suppose an automaton $\mathcal{D}$ is constructed as described above from G, Σ_{o1}, Σ_{o2} and Σ_t. Let $\vec{s}\in\Sigma^{\mathcal{D}^*}$ be a string of events such that $s_1=P_\Sigma(\vec{s})$ and $s_2=\Lambda^{-1}(P_{\Sigma'}(\vec{s}))$ and let $\vec{x}_1=(x_1^1,x_1^2,\epsilon,\epsilon,\epsilon)$ and $\vec{x}_2=(x_2^1,x_2^2,\epsilon,\epsilon,\epsilon)$ be two states in $\mathcal{D}$. In the transition structure of $\mathcal{D}$, $\vec{x}_1\overset{\vec{s}}{\mapsto}_{\mathcal{D}}\vec{x}_2$ if and only if $P_1(s_1)=P_1(s_2)$,

$P_2(s_1) = P_2(s_2)$, $x_1^1 \overset{s_1}{\mapsto}_G x_2^1$ and $x_1^2 \overset{s_2}{\mapsto}_G x_2^2$ subject to $f \notin \Phi(s_2)$.

It is now shown in the theorem below that a marked state $\vec{x}_m \in X_m^{\mathcal{D}}$ is reachable in $\mathcal{D}$ if and only if there exists a pair of strings (s_1, s_2) such that $x_0 \overset{s_1}{\mapsto}_G x_1$, $x_0 \overset{s_2}{\mapsto}_G x_2$, $P_1(s_1) = P_1(s_2)$, $P_2(s_1) = P_2(s_2)$ and $f \in \Phi(s_1)$, but $f \notin \Phi(s_2)$.

Theorem 3: Suppose an automaton $\mathcal{D}$ is constructed as described above from G, Σ_{o1} and Σ_{o2}. A marked state $\vec{x} = (x_1, x_2, \epsilon, \epsilon, \epsilon)$ is reachable in $\mathcal{D}$ if and only if there exists a pair of strings (s_1, s_2) both contained in $\mathcal{L}(G)$ such that $P_1(s_1) = P_1(s_2)$ and $P_2(s_1) = P_2(s_2)$, $x_0 \overset{s_1}{\mapsto}_G x_1$, $x_0 \overset{s_2}{\mapsto}_G x_2$ and $f \in \Phi(s_1)$, but $f \notin \Phi(s_1)$.

It is now shown in Theorem 4 below that there is set of transitions that form a self loop through $\vec{x} = (x_1, x_2, \epsilon, \epsilon, \epsilon)$ in $\mathcal{D}$ if and only if there exists a pair of strings (s_1, s_2) such that $x_1 \overset{s_1}{\mapsto}_G x_1$, $x_2 \overset{s_2}{\mapsto}_G x_2$, $P_1(s_1) = P_1(s_2)$, $P_2(s_1) = P_2(s_2)$ and $f \in \Phi(s_1)$, but $f \notin \Phi(s_2)$. This is shown in the following theorem.

Theorem 4: Suppose an automaton $\mathcal{D}$ is constructed as described above from G, Σ_{o1} and Σ_{o2}. There exists a string $\vec{s}$ and a state $\vec{x} = (x_1, x_2, \epsilon, \epsilon, \epsilon)$ such that $\vec{x} \overset{\vec{s}}{\mapsto}_{\mathcal{D}} \vec{x}$ if and only if there exists a pair of strings (s_1, s_2) such that $P_1(s_1) = P_1(s_2)$ and $P_2(s_1) = P_2(s_2)$, $x_1 \overset{s_1}{\mapsto}_G x_1$, $x_2 \overset{s_2}{\mapsto}_G x_2$ and $f \notin \Phi(s_2)$.

Theorems 3 and 4 can now be used to demonstrate the following theorem.

Theorem 5: Consider the automaton $\mathcal{D}$ constructed from G, Σ_{o1} and Σ_{o2} as introduced above. There exists a string $\vec{s} \in \mathcal{L}_m^{\omega}(\mathcal{D})$ if and only if there exists two infinite strings (s_1, s_2) such that $s_1 \in \mathcal{L}^{\omega}(G)$, $s_2 \in \mathcal{L}^{\omega}(G)$, $f \in \Phi(s_1)$, $f \notin \Phi(s_2)$, $P_1(s_1) = P_1(s_2)$ and $P_2(s_1) = P_2(s_2)$.

It can now be demonstrated that the automaton construction $\mathcal{D}$ can be used to test combined observation diagnosability.

Theorem 6: For the automaton $\mathcal{D}$ constructed from G, Σ_{o1} and Σ_{o2} as introduced above, $\mathcal{L}_m^{\omega}(\mathcal{D}) = \emptyset$ if and only if $\mathcal{L}(G)$ is combined observation diagnosable with respect to the local projections $P_1(\cdot)$ and $P_2(\cdot)$ and the failure f.

Note that there are no bounds placed on the capacities of the r_1 and r_2 queues due to the transition structure of $\mathcal{D}$, but the $r_{1\cap 2}$ queue has a maximum capacity of 1. Also note that because it is assumed that events in Σ_t will always eventually occur, then an event in $\Sigma_{1\cap 2}$ will always eventually occur. Consequently, this bounds the number of $\Sigma_{1\setminus 2}$ and $\Sigma_{2\setminus 1}$ events that could occur between occurrences of $\Sigma_{1\cap 2}$ events. This bound consequently restricts the maximum length of r_1 and r_2 during the operation of $\mathcal{D}$ and hence restricts $\mathcal{D}$ to being finite-state. Therefore, testing if $\mathcal{L}_m^{\omega}(\mathcal{D}) = \emptyset$ is decidable and can be tested by checking $\mathcal{D}$ for a reachable marked state $\vec{x}_m \in X_m^{\mathcal{D}}$ and a string of events $\vec{s}$ such that $\vec{x}_m \overset{\vec{s}}{\mapsto}_{\mathcal{D}} \vec{x}_m$. Therefore, by Theorem 6, the combined observation diagnosability of a system is decidable despite the decentralized nature of the observation actions.

V. Conclusions

A novel distributed diagnosis framework is introduced in this paper where distributed observation systems periodically transmit their observations in batch mode to a centralized coordinator that performs diagnosis. A property called distributed observation diagnosis is introduced to describe systems with failures that can be diagnosed in the above framework. A finite-state automaton construction $\mathcal{D}$ is also introduced that can be used to test if a system G is combined observation diagnosable.

References

[1] E. Athanasopoulou, C.N. Hadjicostis, and K. Rohloff. The distributed diagnosis of discrete-event systems. Preprint.

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

[3] 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.

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

[5] A. Puri, S. Tripakis, and P. Varaiya. Problems and examples of decentralized observation and control for discrete event systems. In B. Caillaud, P. Darondeau, L. Lavagno, and X. Xie, editors, *Synthesis and Control of Discrete Event Systems*. Kluwer Academic Publishers, 2002.

[6] W. Qiu and R. Kumar. Decentralized failure diagnosis of discrete event systems. In *Proc. 7th Workshop on Discrete Event Systems*, September 2004.

[7] P.J. Ramadge. Some tractable supervisory control problems for discrete-event systems modeled by Büchi automata. *IEEE Trans. Auto. Contr.*, 34(1):10–19, 1989.

[8] K. Rohloff. Diagnosis via the unsynchronized periodic communication of distributed observations, 2005. Preprint.

[9] K. Rudie and J.C. Willems. The computational complexity of decentralized discrete-event control problems. *IEEE Trans. Auto. Contr.*, 40(7):1313–1318, 1995.

[10] M. Sampath, S. Lafortune, and D. Teneketzis. Active diagnosis of discrete-event systems. *IEEE Trans. Auto. Contr.*, 43(7):908–929, July 1998.

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

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