Distributed Fault Diagnosis in Discrete Event Systems With Transmission Delay Impairments

Jiwei Wang, Simone Baldi, *Senior Member, IEEE*, Wenwu Yu, *Senior Member, IEEE*, and Xiang Yin, *Member, IEEE*

***Abstract*—This note studies the distributed fault diagnosis problem in partially-observed discrete event systems, where the system is monitored by a group of agents to cooperatively diagnose faults within a finite number of steps. The novelty of this work is the creation of a methodology to verify when the faults can be diagnosed even in the presence of transmission delay impairments. To address this scenario, a new distributed diagnosability condition is proposed, which extends decentralized diagnosability conditions proposed in the literature. Such distributed diagnosability condition is then verified via a novel structure named delay recorder and a new diagnosis function. Theoretical analysis shows that the verification method can successfully determine whether the faults can be diagnosed.**

***Index Terms*—Diagnosability, discrete event systems (DES), distributed fault diagnosis, transmission delay impairments.**

I. Introduction

In recent decades, fault diagnosis for discrete event systems (DES) has attracted increasing attention [1], [2], [3], with the most studied problems being the verification [4], [5] and the synthesis problems [6], [7]. In fault diagnosis of DES, the challenge is to diagnose the occurrence of a fault in finite steps by only observing limited events, while other events including the faults are unobservable. In such partially-observed DES, the fault diagnosis architecture is called centralized when there is only one agent monitoring all observable events [8], [9].

However, as limited coverage and limited communication ability may make a centralized architecture unpractical, multiagent architectures have been proposed, where the system is monitored by a group of agents, each one having partial observation capability [10], [11], [12], [13], [14], [15], [16]. If each agent can share its information with a few neighboring agents, the multiagent architecture is called distributed, otherwise it is decentralized. Decentralized multiagent diagnosis problems have been mostly considered in DES literature, starting from the notion of codiagnosability [10]: codiagnosability is an extension of the single-agent (centralized) diagnosability to a multiagent decentralized scenario, i.e., without information sharing among agents. Related notions have been studied, Yin and Lafortune [11] discussed the connection between codiagnosability and coobservability, Viana and Basilio [12] revisited codiagnosability with a new necessary and sufficient condition in [13], [14], [15], and [16], different notions of codiagnosability and verification methods were proposed, where Keroglou et al. [15] and [16] studied distributed diagnosis under ideal transmission. It should be noted that ideal transmission allows each agent to get information with no delay, converging to a centralized scenario.

The distributed multiagent diagnosis problem is largely open, and in particular no framework exists to address the inevitable transmission delay impairments associated with information sharing. Hence, this article proposes a new diagnosis framework to handle such an issue in the DES. The framework is able to account for restricted communication among agents, and it bridges the centralized, the distributed, and the decentralized architectures in a unified way; in fact, the framework we propose comprises the decentralized scenario as the transmission impairments increase and the centralized one as the impairments vanish.

The main difficulties in developing this framework lie in dealing with the uncertain delays arising from transmitting and processing the information. Notably, as some information may not contribute to fault diagnosis, a method should be put in place to identify those events whose delays need to be recorded. Novel methods and structures are put forward, which form the main contributions of this article. We propose a new condition for distributed fault diagnosis, namely K^T-codiagnosability (cf. Definition 3), which extends in a natural way the state-of-the-art notion of codiagnosability within K steps, i.e., K-codiagnosability (cf. Definition 2). We propose a novel structure, named as delay recorder (cf. Algorithm 1), to record the delays required for diagnosis; a new diagnosis function is proposed, with which K^T-codiagnosability is verified (cf. Theorem 1). Here, T refers to the transmission efficiency. Our framework comprises the decentralized K-codiagnosability as T decreases (i.e., impairments increase), and the centralized K-diagnosability as T increases (i.e., the impairments vanish). Summarizing, this study proposes the first unified framework for distributed fault diagnosis in the DES with transmission delay impairments.

The rest of this article is organized as follows. Section II describes the partially-observed DES. Section III proposes the key notion of K^T-codiagnosability. In Section IV, the delay recorder and diagnosis function are proposed to accomplish verification. Finally, Section V concludes this article.

II. Preliminaries

Let us recall the basic formalism of automata, used to model DES. Consider the finite event set E in DES as an alphabet, so that the

Manuscript received 17 December 2022; revised 2 September 2023; accepted 10 February 2024. Date of publication 23 February 2024; date of current version 30 July 2024. This work was supported in part by the National Key R&D Program of China under Grant 2022YFE0198700, and in part by the National Natural Science Foundation of China under Grant 62150610499, Grant 62073074, Grant 62073076, Grant 62233004, and Grant 62173226. Recommended by Associate Editor J. Komenda. *(Corresponding authors: Simone Baldi; Wenwu Yu.)*

Jiwei Wang is with the School of Cyber Science and Engineering, Southeast University, Nanjing 210096, China (e-mail: jwwang@seu.edu.cn).

Simone Baldi and Wenwu Yu are with the School of Mathematics, Southeast University, Nanjing 210096, China (e-mail: simonebaldi@seu.edu.cn; wwyu@seu.edu.cn).

Xiang Yin is with the Department of Automation and Key Laboratory of System Control and Information Processing, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: yinxiang@sjtu.edu.cn).

Digital Object Identifier 10.1109/TAC.2024.3369510

concatenation of a string of words in the alphabet can be viewed as a finite sequence of events in E. A *language* is a set of event strings, formed from the events in E. Let E^* be the set of all finite strings over E. Denote the length of a string s as $|s|$, and let ϵ be the empty string with $|\epsilon| = 0$. The *prefix-closure* of a language L is defined as $\overline{L} = \{s \in E^* | \exists t \in E^*, \text{ s.t. } st \in L\}$ and L is *prefix-closed* if $L = \overline{L}$. We assume to work with *live* languages L: $\forall s \in L, \exists e \in E, \text{s.t. } se \in L$, that is, any string in L can be extended to an arbitrary length.

Automata are a common framework for manipulating languages. Let us consider a finite automaton

$$G = (X, E, \alpha, X_0) \tag{1}$$

where X is the set of finite states; E is the set of finite events; $\alpha : X \times E^* \to 2^X$ is the transition function that describes the transition of an event string; and $X_0 \subseteq X$ is the set of possible initial states. The language generated by G from state $x \in X$ is denoted by $\mathcal{L}(x, G) = \{s \in E^* | \alpha(x, s)!\}$, where ! means that the string s "is defined", i.e., it can occur starting from state x. If $x \in X_0$, we simply denote $\alpha(x_0, s)$ as $\alpha(s)$ and $\mathcal{L}(x_0, G)$ as $\mathcal{L}(G)$. Given a set of states $\iota \subseteq X$, we define the set of accessible states of ι as $\mathcal{A}^G(\iota) = \{x' \in X \big| \exists x \in \iota, \exists s \in \mathcal{L}(x, G), \text{ s.t. } x' \in \alpha(x, s)\}$.

In a partially-observed DES, the event set E is divided into the observable events E_o and the unobservable events E_{uo}. A projection operator $P_{E_o} : E^* \to E_o^*$ is used to obtain the observation of an event string as follows: $\forall s \in \mathcal{L}(G), \forall e \in E : \alpha(se)!$,

$$P_{E_o}(\epsilon) = \epsilon, P_{E_o}(se) = \begin{cases} P_{E_o}(s)e, & \text{if } e \in E_o \\ P_{E_o}(s), & \text{if } e \notin E_o. \end{cases} \tag{2}$$

Intuitively, $P_{E_o}(s)$ shows the observed events for a trajectory $s \in \mathcal{L}(G)$. The operator P_{E_o} can also handle a set of event string, that is, $\forall S \subseteq \mathcal{L}(G),\ P_{E_o}(S) = \{s \in E_o^* | \exists s' \in S, \text{ s.t. } s = P_{E_o}(s')\}$. Based on the projection operator P_{E_o}, consider an operator $\zeta^n_{E_o}$: $\forall s \in \mathcal{L}(G)$,

$$\zeta^n_{E_o}(s) = \{s'' \in \overline{s} | \exists s' \in \overline{s} : |s'| \geq |s| - n, \text{s.t.} P_{E_o}(s'') = P_{E_o}(s')\}.$$

Intuitively, $\zeta^n_{E_o}(s)$ collects all prefixes of s with the same observation as a trajectory s'.

To embed the information of observable events into the system model and determine the indistinguishable states, we make use of the M-machine w.r.t. G and E_o as [17] and [18]

$$\mathcal{M}_{E_o}(G) = (Z, E \cup \{\epsilon\}, \delta, Z_0) \tag{3}$$

where $Z \subseteq X \times X$ is the set of states and $Z_0 = X_0 \times X_0$ is the set of initial states. For any $(x_1, x_2) \in Z, e \in E$, the transition function $\delta : Z \times E \cup \{\epsilon\} \to 2^Z$ is defined as follows.

1) If $e \in E_o \wedge \alpha(x_1, e)! \wedge \alpha(x_2, e)!$, then, $\delta((x_1, x_2), e) = \{(x_1', x_2') \mid x_1' \in \alpha(x_1, e), x_2' \in \alpha(x_2, e)\}$.
2) If $e \notin E_o \wedge \alpha(x_1, e)!$, then, $\delta((x_1, x_2), e) = \{(x_1', x_2) \mid x_1' \in \alpha(x_1, e)\}$.
3) If $e \notin E_o \wedge \alpha(x_2, e)!$, then, $\delta((x_1, x_2), \epsilon) = \{(x_1, x_2') \mid x_2' \in \alpha(x_2, e)\}$.

Intuitively, the definitions of α and δ are such that $\mathcal{L}(G) = \mathcal{L}(\mathcal{M}_{E_o}(G))$. In addition, we have $\forall s \in \mathcal{L}(G), \delta(s) = \{(x, x') \in Z \mid x \in \alpha(s), x' \in \alpha(s') : P_{E_o}(s') = P_{E_o}(s)\}$, which implies that for any $(x, x') \in Z$, x and x' are indistinguishable with the observation ability E_o. In the following, we use the notation $I_1(x, x') = x$ and $I_2(x, x') = x'$ to indicate the first and the second state component of $(x, x') \in Z$.

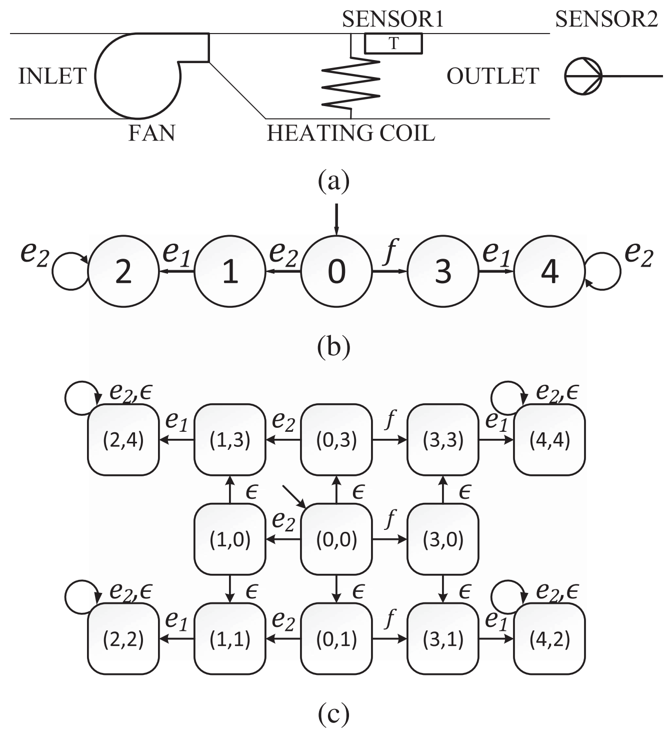

Fig. 1. Air heating unit, the model G of its start-up process and the M-machine $\mathcal{M}_{E_o'}(G)$ with $E_o' = \{e_1\}$. (a) Air heating unit. (b) G. (c) $\mathcal{M}_{E_o'}(G)$.

A. Illustrative Example

To illustrate the key concepts, we present throughout this work a few examples inspired by an air heating unit start-up scenario, cf. Fig. 1(a). At start-up, under healthy conditions, the fan creates an air flow heated by the heating coil. The air flow blows the heat away from the coil so that a desired temperature is reached at an equilibrium. But in some start-up scenarios, the fan may fail to turn ON and we need to diagnose the fault to avoid coil overheating. The system is monitored by the following two sensors: 1) a temperature sensor close to the coil and 2) an air flow sensor at the outlet. Denote the event observed by sensor 1 as e_1 (desired temperature is reached) and the event observed by sensor 2 as e_2 (flow rate is regular). The fault, obviously unobservable by any sensor, is denoted as f.

Example 1: (System model). We model the start-up of the air heating unit as the automaton G in Fig. 1(b), where the initial state $x_0 = \{0\}$ means that the system is OFF initially. The branch on the left of state 0 represents the healthy functioning. The air flow is regular (in state 1), so that after some time the desired temperature is reached (in state 2). The branch starting on the right of state 0 represents the scenario that the fan does not start, which may be due to an unobservable fault (in state 3), leading to overheating detected via sensor 1 (in state 4). However, it is possible that the fan simply did not start timely (e.g., due to blockage in the flow channel, or wear), and after some time, e_2 may occur, detected by sensor 2. Using the automaton formalism, we have that when the system is in state 0, the only events that can occur are f and e_2, that is, $\alpha(0, f)!$ and $\alpha(0, e_2)!$. For the string $fe_1e_2e_2$ generated by G, let $E_o = \{e_1, e_2\}$. Then, $P_{E_o}(fe_1e_2e_2) = e_1e_2e_2$, $\zeta^0_{E_o}(fe_1e_2e_2) = \{fe_1e_2e_2\}$ and $\zeta^3_{E_o}(fe_1e_2e_2) = \{\epsilon, f, fe_1, fe_1e_2, fe_1e_2e_2\}$. To illustrate the M-machine, consider an observable event set $E_o' = \{e_1\}$. Then, we have $\mathcal{M}_{E_o'}(G) = (Z', E \cup \{\epsilon\}, \delta', Z_0')$ shown in Fig. 1(c). For $fe_1e_2e_2 \in \mathcal{L}(\mathcal{M}_{E_o'}(G))$, we have $\delta'(fe_1e_2e_2) = \{(4, 4), (4, 2)\}$,

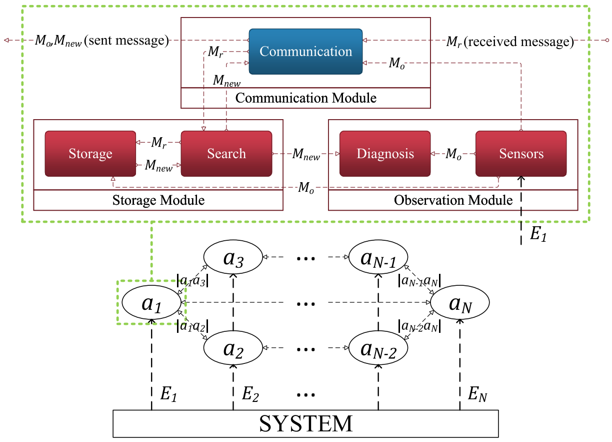

Fig. 2. Distributed observation architecture (lower) and relations between the three modules of each agent (upper).

indicating that states 4 and 2 are indistinguishable when we rely only on the observation of e_1. □

III. Distributed Diagnosability

The notion of codiagnosability [10] was proposed as the basic property to handle decentralized fault diagnosis, i.e., without information sharing between agents. We provide a distributed extension, called K^T-codiagnosability, when information sharing between agents is allowed (possibly subject to transmission impairments). Before this, we discuss the distributed observation architecture and ambiguities arising from partial observation and transmission impairments.

A. Distributed Observation

Let the system under consideration be monitored by a set of agents $A = \{a_1, a_2, \ldots, a_N\}(N \in \mathbb{N}^+)$ with corresponding events in $\{E_1, E_2, \ldots, E_N\}$ such that $E_o = E_1 \cup \cdots \cup E_N$. Each agent can share its information with some of the other agents according to a weighted connected graph $C_A = (V_A, W_A)$ consisting of a set of vertices $V_A = \{a_1, \ldots, a_N\}$ representing the agents, and a set of undirected weighted edges $W_A \subseteq V_A \times V_A$ representing the transmission links among neighboring agents; the nonnegative weight of each edge is related to a transmission delay as specified later. Denote the length of the path between two vertices as the sum of the weights along the path. Then, for any two agents a_i, a_j, we define their distance $|a_i a_j|$ as the minimum length between them.

With the distributed structure above, we now consider a simple communication protocol between agents. The following three modules are required for each agent: 1) communication; 2) storage; and 3) observation modules in Fig. 2:

1) *Communication module:* this module forwards M_r (message received from neighbours) to the storage module, and sends M_{new} (new message from storage module) and M_o (message from observation module) to the neighbours.
2) *Storage module:* this module stores M_o from the observation module, and avoids that the occurrence of a certain event is recorded multiple times, in fact, M_r is stored as M_{new} only if it is not already in the storage set.
3) *Observation module:* in this module, a new observed event from sensors receives a timestamp and becomes M_o; the module also performs diagnosis by processing M_o, M_{new} with a diagnoser [1] or an observer [2].

The operations of communication and storage are expected to introduce delays between the observation of an event by one of the agents and the reception of the same event by the other agents. Nevertheless, as C_A is a connected graph, the message that an event is observed (and its timestamp) will reach all other agents, possibly with some delay. For better readability and in line with the literature (e.g., [10], [11], [12], [13], [14]), we will simply analyze two agents $a_i (i \in \{1, 2\})$. All the results in this work can be extended to more agents, at the price that more cases should be analyzed.

Note that the messages processed by each agent $a_i (i \in \{1, 2\})$ have two sources as follows: 1) M_o from the observation module and 2) M_{new} from the storage module. Thus, the set of all observable events E_o can be partitioned into E_i and $E_o \backslash E_i$, where the occurrence of the events in E_i is received with no delay, while the occurrence of the events in $E_o \backslash E_i$ is received from the storage module with some delay.

We now introduce a coefficient $T > 0$ related to transmission efficiency, where $T = 1$ indicates a nominal efficiency and $T < 1$ indicates that the efficiency degrades. The distance between agents is $|a_1 a_2| \geq 0$ and we represent the transmission delays as follows: if another agent a_k $(k \neq i)$ observes an event $e \notin E_i$, a_i will receive the observation e with a delay of no more than $\lceil \frac{|a_1 a_2|}{T} \rceil$ steps ($\lceil \cdot \rceil$ rounds the element to the nearest integer towards infinity). Note that $T \ll 1$ degrades to a decentralized setting, whereas $T \gg 1$ converges to a centralized setting where each agent can monitor *all* observable events with delay of no more than one step.

B. K^t-Codiagnosability

As a starting point for fault analysis in the distributed setting, we recall state-of-the-art notions for centralized and decentralized fault diagnosis.

Let $G = (X, E, \alpha, x_0)$ be the system model and f be the fault events we intend to diagnose. Let K be the maximum number of steps allowed from the occurrence of a fault to its diagnosis. To diagnose the faults within K steps, a structure of step counter $\Delta : \mathcal{L}(G) \to \{-1, 0, 1, \ldots, K\}$ is used to count the number of steps in an event string after a fault occurs: $\forall s \in \mathcal{L}(G), \forall e \in E : \alpha(se)! \Rightarrow \Delta(\epsilon) = -1, \Delta(se) =$

$$\begin{cases} \Delta(s), & \text{if } [\Delta(s) = -1 \wedge e \neq f] \vee [\Delta(s) = K] \\ \Delta(s) + 1, & \text{if } [\Delta(s) = -1 \wedge e = f] \vee [0 \leq \Delta(s) < K] \end{cases} \tag{4}$$

where -1 means no fault happens. By means of Δ, the literature has introduced the notions of K-diagnosability and K-codiagnosability.

Definition 1. (K-diagnosability [19]): For $K \in \mathbb{N}$, the live language $\mathcal{L}(G)$ is K-diagnosable w.r.t. f if $\forall s \in \mathcal{L}(G) : \Delta(s) = K$,

$$\forall s' \in \mathcal{L}(G) : P_{E_o}(s') = P_{E_o}(s), \Delta(s') \neq -1. \tag{5}$$

Definition 2. (K-codiagnosability [20]): For $K \in \mathbb{N}$, the live language $\mathcal{L}(G)$ is K-codiagnosable w.r.t. f if $\forall s \in \mathcal{L}(G) : \Delta(s) = K$,

$$\exists i \in \{1, 2\}, \text{s.t.} \forall s' \in \mathcal{L}(G) : P_{E_i}(s') = P_{E_i}(s), \Delta(s') \neq -1. \tag{6}$$

Obviously, K-diagnosability is a centralized notion as a single monitors all observable events. In K-codiagnosability, f can be diagnosed by either a_1 or a_2 unambiguously within K steps, without any communication between agents. Unfortunately, the following example shows that some faults may go undetected in the absence of communication.

Example 2. (Limits of K-codiagnosability): For the system in Fig. 1(b), we consider a_1 with observation ability $E_1 = \{e_1\}$, and a_2 with observation ability $E_2 = \{e_2\}$. Since $P_{E_1}(f e_1 e_2 e_2) = P_{E_1}(e_2 e_1 e_2) = e_1$ and $P_{E_2}(f e_1 e_2 e_2) = P_{E_2}(e_2 e_1 e_2) = e_2 e_2$, i.e., faulty and healthy strings are indistinguishable, it is impossible for a_1 or a_2 to determine within $K = 3$ steps if the fault f has occurred or not. We conclude that, when $K = 3$, $\mathcal{L}(G)$ is not K-codiagnosable w.r.t. f. □

Intuitively, a fault that goes undetected in the absence of communication may become detectable if communication among agents is allowed (cf. Example 3). This means that K-codiagnosability is restrictive and an appropriate extension is required, which is the key definition in this article.

Definition 3. (K^T-codiagnosability): For $K \in \mathbb{N}$ and $T > 0$, the live language $\mathcal{L}(G)$ is K^T-codiagnosable w.r.t. f if $\forall s \in \mathcal{L}(G) : \Delta(s) = K$,

$$\exists i \in \{1, 2\}, \text{ s.t. } \forall s' \in \mathcal{L}(G) : P_{E_i}(s') = P_{E_i}(s) \wedge$$

$$P_{E_o}(\zeta_{E_o\setminus E_i}^{\lceil\frac{|a_1a_2|}{T}\rceil}(s')) \cap P_{E_o}(\zeta_{E_o\setminus E_i}^{\lceil\frac{|a_1a_2|}{T}\rceil}(s)) \neq \emptyset, \Delta(s') \neq -1. \quad (7)$$

Intuitively, if (7) holds, then, a_i can timely observe or receive all key events to determine the occurrence of f. In other words, a_i is capable of consistently distinguishing a fault string (s satisfying $\Delta(s) = K$) from a normal string (s satisfying $\Delta(s) = -1$), despite the imperfect observation caused by delay. As T increases, the strings that cannot be distinguished from the string $s : \Delta(s) = K$ become less and less, that is, (7) gets weaker and weaker. As expected, K^T-codiagnosability $\Rightarrow K^{T'}$-codiagnosability when $T \leq T'$ (higher transmission efficiency improves diagnosis ability).

Example 3. (K^T-codiagnosability): Consider the same system and agents as Example 2. Suppose $|a_1a_2| = 2$, $T' = 2$, then, a_1 will receive the occurrence of e_2 in no more than $\lceil\frac{|a_1a_2|}{T'}\rceil = 1$ step. When e_2e_1 occurs, the occurrence of e_2 will be received by a_1 before e_1, but no e_2 will be received by a_1 before the observation of e_1 when fe_1 occurs. This implies that e_2e_1 and fe_1 are distinguishable, i.e., the fault can be diagnosed by a_1. Indeed, Definition 3 gives $P_{E_1}(fe_1) = e_1$, $P_{E_o}(\zeta^1_{E_2}(fe_1)) = \{\epsilon\}$, and each string $s' \in \{s|P_{E_1}(s) = e_1 \wedge P_{E_o}(\zeta^1_{E_2}(s)) \cap \{\epsilon\} \neq \emptyset\} = \{fe_1, fe_1e_2\}$ satisfies $\Delta(s') \geq 0$. We conclude that $\mathcal{L}(G)$ is $K^{T'}$-codiagnosable w.r.t. f when $T' = 2$ and $K \geq 1$. Next, suppose $T = 1$, so that a_1 will receive the occurrence of e_2 in no more than $\lceil\frac{|a_1a_2|}{T}\rceil = 2$ steps. In this case, we know that $fe_1e_2e_2$ and e_2e_1 are indistinguishable since e_2 may not be received before e_1. Correspondingly, the string set $\{s \mid P_{E_1}(s) = P_{E_1}(fe_1e_2e_2) \wedge P_{E_o}(\zeta^2_{E_2}(s)) \cap P_{E_o}(\zeta^2_{E_2}(fe_1e_2e_2)) \neq \emptyset\} = \{e_2e_1, fe_1e_2, fe_1e_2e_2, \dots\}$ and $\Delta(e_2e_1) = -1$. That is, the fault may not be diagnosed by a_1 when $T = 1$ and $K = 3$. Nevertheless, Example 6 will show that K^T-codiagnosability is satisfied when $T = 1$ and $K = 3$, as the fault can be diagnosed by a_2. □

Remark 1. (Relations between K-codiagnosability, K^T-codiagnosability and K-diagnosability): Obviously, (6) $\Rightarrow$ (7), that is, if K-codiagnosability holds, then, K^T-codiagnosability holds for any T. From (7) and the definition of ζ, we have $P_{E_o}(s') = P_{E_o}(s) \Leftrightarrow P_{E_o}(\zeta^0_{E_o\setminus E_i}(s')) = P_{E_o}(\zeta^0_{E_o\setminus E_i}(s)) \Rightarrow P_{E_i}(s') = P_{E_i}(s) \wedge P_{E_o}(\zeta_{E_o\setminus E_i}^{\lceil\frac{|a_1a_2|}{T}\rceil}(s')) \cap P_{E_o}(\zeta_{E_o\setminus E_i}^{\lceil\frac{|a_1a_2|}{T}\rceil}(s)) \neq \emptyset$ for any T. We obtain that (7) $\Rightarrow$ (5), that is, if K^T-codiagnosability holds for any T, then, K-diagnosability holds. Hence, we conclude the following:

K-codiagnosability$\Rightarrow K^T$ -codiagnosability $\Rightarrow$ K-diagnosability.

IV. Verification of Distributed Diagnosability

In general, it is impossible to determine if a fault can be diagnosed by analyzing each event string as in Example 3. It is necessary to embed the delay information into the automaton and develop a feasible method to verify K^T-codiagnosability. This is done by linking the system states to fault events and by building a delay recorder structure to handle the delays.

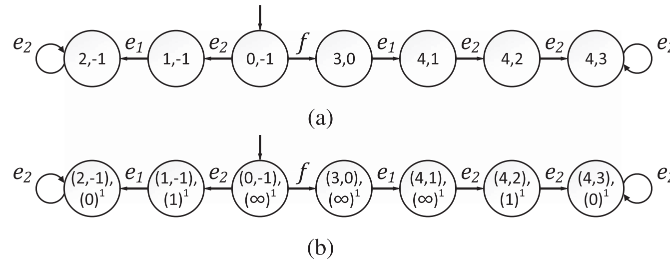

Fig. 3. Fault automaton $\hat{G}$ and delay recorder $\mathcal{R}'_1$. (a) $\hat{G}$. (b) $\mathcal{R}'_1$.

A. Delay Recorder

In diagnosis, it is crucial to observe the events that help to distinguish the faulty from the healthy strings. A delay recorder aims to register the delays of these events correctly. Note that recording all the delays can be deleterious for verification, which will be more clear in Example 4.

Motivated by the step counter Δ in (4), a structure of fault automaton is constructed from (1) to count the number of steps after a fault happens [11]

$$\hat{G} = (\hat{X}, E, \hat{\alpha}, \hat{X}_0) \quad (8)$$

where $\hat{X} = X \times \{-1, 0, 1, \dots, K\}$ includes the state in X and the fault counting component, i.e., $\hat{x} = (x, |\hat{x}|_f) \in \hat{X}$ where $|\hat{x}|_f$ indicates the number of steps after a fault occurs, as calculated in [11]. The transition function $\hat{\alpha} : \hat{X} \times E \to 2^{\hat{X}}$ is defined as follows: for any $\hat{x} = (x, |\hat{x}|_f) \in \hat{X}$ and $e \in E$ satisfying $\alpha(x, e)!$, we have $\hat{\alpha}((x, |\hat{x}|_f), e) = \{(x', |\hat{x}|_f + v)|x' \in \alpha(x, e)\}$, where $|\hat{x}|_f \in \{-1, 0, 1, \dots, K\}$ and v is defined by

$$v = \begin{cases} 0, & \text{if } [|\hat{x}|_f = -1 \wedge e \neq f] \vee [|\hat{x}|_f = K] \\ 1, & \text{if } [|\hat{x}|_f = -1 \wedge e = f] \vee [0 \leq |\hat{x}|_f < K]. \end{cases}$$

The set of initial states is $\hat{X}_0 = \{(x_0, -1) \mid x_0 \in X_0\}$. Obviously, $\mathcal{L}(\hat{G}) = \mathcal{L}(G)$. Let us consider the M-machine w.r.t. $\hat{G}$ and E_o, that is, $\mathcal{M}_{E_o}(\hat{G}) = (Z, E \cup \{\epsilon\}, \delta, Z_0)$. For each state $z \in Z \subseteq \hat{X} \times \hat{X}$, we have that $I_1(z), I_2(z) \in \hat{X}$: thus, we can use $|I_1(z)|_f$ and $|I_2(z)|_f$ to represent the fault counting value. We denote the "confusing state" subset as

$$Z^C = \{z : |I_1(z)|_f = K \wedge |I_2(z)|_f = -1\} \subseteq Z. \quad (9)$$

We are now in the position to explain how to handle delays. For agent $a_i (i \in \{1, 2\})$, the delay recorder to determine the delays to be recorded is defined as an automaton

$$\mathcal{R}_i = (X_i, E, \alpha_i, X_{i0}) \quad (10)$$

where each state $x_i = (\hat{x}, (|x_i|_i^{j_i})^{j_i}) \in X_i \subseteq \hat{X} \times (\{0, 1, \dots, \lceil\frac{|a_1a_2|}{T}\rceil, \infty\})^{j_i} (j_i \in \{1, \dots, n_i\})$ contains the following two components: 1) the state in $\hat{X}$ and 2) the delay value, denoted by $|x_i|_i^{j_i}$. Here, n_i is the number of the delay value we need to record. The transition function $\alpha_i : X_i \times E \to 2^{X_i}$ and the set of initial states X_{i0} are built as in Algorithm 1.

Algorithm 1 consists of two parts: the first part (lines 1–20) marks n_i sets of transitions, utilized to build the delay recorders with the `RECORD` procedure in the second part (lines 21–29). All the procedures are listed in Algorithm 2. The procedure `FORWARD1`(ι, Y_i^j) collects the transition sets for Y_i^j, that, only containing the events in $E_o\setminus E_i$, cannot be distinguished under the observation ability E_o. The procedure `FORWARD2`(Y_i^j) explores new transition sets; lines 3–20 check which transition set in Y_i^j is necessary to be marked. The procedure `RECORD` appends the delay ∞ to the subsequent states of the marked transitions,

Algorithm 1: The Construction of the Delay Recorder $\mathcal{R}_i$.

Input: $\hat{G}=(\hat{X},E,\hat{\alpha},\hat{X}_0)$, $\lceil\frac{|a_1a_2|}{T}\rceil$, $Z^C_{E_i}$, E_o, $E_i (i\in\{1,2\})$;
Output: $\mathcal{R}_i=(X_i,E,\alpha_i,X_{i0})$;
$T^{ini},T_i^1,X_i^1,Y_i^1,X_{i0},X_i\leftarrow\emptyset$; $m\leftarrow 1$;
FORWARD1 $(\hat{X}_0,X_i^1,Y_i^1)$; FORWARD2(Y_i^1); $n\leftarrow m$;
for $l\in\{1,\dots,n\}$ **do**
  $I\leftarrow\bigcup_{y\in Y_i^{l-n+m}}\{\{\hat{x}'\in\hat{X}|\exists\hat{x}\in\hat{X}:(\hat{\alpha}(\hat{x},e)=\hat{x}')\in y\}\}$;
  if $\forall(\hat{x}^k,\hat{x}^{-1})\in Z^C_{E_i}$, $[\forall\iota,\iota'\in I:\iota\neq\iota',\hat{x}^k\notin\mathcal{A}^{\hat{G}}(\iota)$, $\hat{x}^{-1}\notin\mathcal{A}^{\hat{G}}(\iota')]\wedge[\hat{x}^k\notin X_i^{l-n+m}\vee\hat{x}^{-1}\notin X_i^{l-n+m}]$ **then**
    $T_i^{l-n+m}\leftarrow T_i^m$; $X_i^{l-n+m}\leftarrow X_i^m$; $Y_i^{l-n+m}\leftarrow Y_i^m$;
    $m\leftarrow m-1$;
  end if
end for
$n\leftarrow m$; $n'\leftarrow 1$; $n_i\leftarrow 1$;
for $j\in\{2,3,\dots,n\}$ **do**
  if $\forall l\in\{1,\dots,n_i\}$, $\exists y\in Y_i^j$, s.t. $\forall y'\in Y_i^l, y\not\subseteq y'$ **then**
    for $l'\in\{0,\dots,n'-1\}$ **do**
      if $\forall y\in Y_i^{n'-l'}$, $\exists y'\in Y_i^j$, s.t. $y\subseteq y'$ **then**
        $T_i^{n'-l'}\leftarrow T_i^{n_i}$; $Y_i^{n'-l'}\leftarrow Y_i^{n_i}$; $n_i\leftarrow n_i-1$;
      end if
    end for
    $n_i\leftarrow n_i+1$; $n'\leftarrow n_i$; $T_i^{n_i}\leftarrow T_i^j$; $Y_i^{n_i}\leftarrow Y_i^j$;
  end if
end for
for $\hat{x}_0\in\hat{X}_0$ **do**
  $X_{i0}\leftarrow X_{i0}\cup\{(\hat{x}_0,(\infty)^1)\}$; $X_i\leftarrow X_i\cup\{(\hat{x}_0,(\infty)^1)\}$;
  RECORD$((\hat{x}_0,(\infty)^1),\mathcal{R}_i)$;
  for $j\in\{2,\dots,n_i\}$ **do**
    $X_{i0}\leftarrow X_{i0}\cup\{(\hat{x}_0,(0)^j)\}$; $X_i\leftarrow X_i\cup\{(\hat{x}_0,(0)^j)\}$;
    RECORD$((\hat{x}_0,(0)^j),\mathcal{R}_i)$;
  end for
end for
Return $\mathcal{R}_i$;
procedure FORWARD1ι,X_i^j,Y_i^j
  $X^{tem}\leftarrow\{\hat{x}\in\hat{X}|\exists\hat{x}'\in\iota,s\in(E\backslash E_o)^*$, s.t. $\hat{x}\in\hat{\alpha}(\hat{x}',s)\}$;
  if $X^{tem}\not\subseteq X_i^j$ **then**
    $X_i^j\leftarrow X_i^j\cup X^{tem}$;
    for $e\in E_o:\exists\hat{x}\in X^{tem}$, s.t. $\hat{\alpha}(\hat{x},e)!$ **do**
      if $e\in E_i$ **then**
        FORWARD1$(\{\hat{x}\in X_i|\exists\hat{x}'\in X^{tem}$, s.t. $\hat{x}\in\hat{\alpha}(\hat{x}',e)\},X_i^j,Y_i^j)$;
      else if $e\in E_o\backslash E_i$ **then**
        $Y_i^j\leftarrow Y_i^j\cup\{\{\hat{\alpha}(\hat{x}',e)=\hat{x}|\exists\hat{x}'\in X^{tem}$, $\hat{x}\in\hat{X}$, s.t. $\hat{x}\in\hat{\alpha}(\hat{x}',e)\}\}$;
      end if
    end for
  end if
end procedure
procedure FORWARD2Y_i^j
  for $y\in Y_i^j$ **do**
    $X'\leftarrow\{\hat{x}'\in\hat{X}|\exists\hat{x}\in\hat{X}:(\hat{\alpha}(\hat{x},e)=\hat{x}')\in y\}$;
    if $[\exists(\hat{x}^k,\hat{x}^{-1})\in Z^C_{E_i}$, s.t. $\hat{x}^k,\hat{x}^{-1}\in\mathcal{A}^{\hat{G}}(X')]\wedge[y\notin T^{ini}]$ **then**
      $T^{ini}\leftarrow T^{ini}\cup y$; $m\leftarrow m+1$;
      $T_i^m\leftarrow y$; $X_i^m\leftarrow\emptyset$; $Y_i^m\leftarrow\emptyset$;
      FORWARD1(X',X_i^m,Y_i^m); FORWARD2(Y_i^m);
    end if
  end for
end procedure

Algorithm 2: The RECORD Procedure in Algorithm 1.

procedure RECORD$(\hat{x},(u)^j),\mathcal{R}_i$
  for $e\in E:\hat{\alpha}(\hat{x},e)!$ **do**
    for $\hat{x}'\in\hat{\alpha}(\hat{x},e)$ **do**
      if $(\hat{\alpha}(\hat{x},e)=\hat{x}')\in T_i^j$ **then**
        $u'\leftarrow\infty$;
      else if $u=\infty\wedge e\in E_o\backslash E_i$ **then**
        $u'\leftarrow\lceil\frac{|a_1a_2|}{T}\rceil$;
      else if $u>0$ **then**
        $u'\leftarrow u-1$;
      else if $u=0$ **then**
        $u'\leftarrow 0$;
      end if
      Add $\alpha_i((\hat{x},(u)^j),e)=(\hat{x}',(u')^j)$ to $\mathcal{R}_i$;
      if $(\hat{x}',(u')^j)\notin X_i$ **then**
        $X_i\leftarrow X_i\cup\{(\hat{x}',(u')^j)\}$;
        RECORD$((\hat{x}',(u')^j),\mathcal{R}_i)$;
      end if
    end for
  end for
end procedure

and records the delay of the transitions that "leave" these states marked by ∞.

As $\mathcal{R}_i$ is built from $\hat{G}$, we have $\mathcal{L}(\mathcal{R}_i)=\mathcal{L}(\hat{G})$. Note that $\mathcal{R}_i$ records the delay of the events in $E_o\backslash E_i$ that help to distinguish the pair of system trajectories s and s' satisfying $P_{E_i}(s)=P_{E_i}(s')$ and $\Delta(s)=K,\Delta(s')=-1$. This is needed to verify K^T-codiagnosability, as it will be clear in the next section.

Example 4. (The importance of a delay recorder): For the system G in Fig. 1(b), the corresponding fault automaton $\hat{G}$ is shown in Fig. 3(a). With the observable event set $E_1=\{e_1\}$, we obtain the M-machine $\mathcal{M}_{E_1}(\hat{G})$ and the confusing state subset $Z^C_{E_1}=\{((4,3),(2,-1))\}$. Next, using $\hat{G}$, $|a_1a_2|=2$, $T'=2$, $Z^C_{E_1},E_o$ and E_1, we run Algorithm 1 to obtain the delay recorder $\mathcal{R}'_1$ shown in Fig. 3(b). From $\mathcal{R}'_1$, one can see that only the delay of $\mathbf{e_2}$ (denoted with bold) in $fe_1\mathbf{e_2}e_2$ and $\mathbf{e_2}e_1e_2$ are recorded. The delay value "0" in $((4,3),(0)^1)$ and $((2,-1),(0)^1)$ indicates that the occurrence of these $\mathbf{e_2}$ that help to distinguish $fe_1e_2e_2$ and $e_2e_1e_2$ have been received, which implies that the fault can be diagnosed by a_1, as shown in Example 3 with $T'=2$. Nevertheless, without a delay recorder, a naive strategy could be to record each delay of $\mathbf{e_2}$, that is, $fe_1\mathbf{e_2}\mathbf{e_2}$ and $\mathbf{e_2}e_1\mathbf{e_2}$. Unfortunately, by doing this, one would obtain $((4,3),(1)^1)$ and $((2,-1),(1)^1)$, where the delay value "1" implies that we cannot determine if e_2 has been received by a_1, leading to a trouble for verification. □

Remark 2. (Complexity analysis): Like existing diagnosis algorithms, Algorithm 1 relies on the construction of observers (or diagnosers). This operation is known from the literature having worst-case exponential complexity $O(2^{2^n})$ [2], where n is the number of states. However, this worst-case complexity is rarely reached, and recent studies have shown that, for deterministic automata, the average state size of diagnosers is $O(n^{0.77\log k+0.63})$ [21], where k is the number of events.

B. Verification of K^t-Codiagnosability

Using all structures introduced before, we now initiate the verification process of K^T-codiagnosability.

Let $\hat{G}$ in (8) be the fault automaton built from G in (1). We first consider the delay value of a_i ($i\in\{1,2\}$). To run Algorithm 1, we

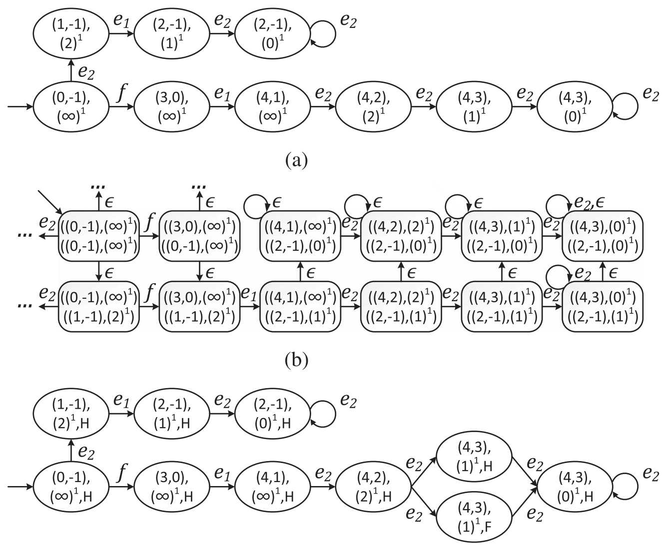

Fig. 4. Part of M-machine $\mathcal{M}_{E_1}(\mathcal{R}_1)$ that may include the fault states and the reconstructed automaton $\hat{\mathcal{R}}_1$. (a) $\mathcal{R}_1$. (b) Part of $\mathcal{M}_{E_1}(\mathcal{R}_1)$. (c) $\hat{\mathcal{R}}_1$.

built $\mathcal{M}_{E_i}(\hat{G})$ to get the state subset $Z^C_{E_i}$, and then, the delay recorder $\mathcal{R}_i$ in (10) is obtained. Aiming to determine the fault states that a_i cannot diagnose even with the received message, we further build the M-machine $\mathcal{M}_{E_i}(\mathcal{R}_i) = (Z_i, E \cup \{\epsilon\}, \delta_i, Z_{i0})$. As $\mathcal{M}_{E_i}(\mathcal{R}_i)$ only contains the delay information of a_i, we need to run Algorithm 1 again with E_k ($k \in \{1,2\}, k \neq i$) to obtain the delay information of a_k. Considering that the input of Algorithm 1 should be a fault automaton, we need reconstruct $\mathcal{M}_{E_i}(\mathcal{R}_i)$ to be a fault automaton with the delay information of a_i. To this end, we remove the second component of each state in Z_i and the empty event ϵ in event set E, constructing the automaton

$$\hat{\mathcal{R}}_i = (\hat{X}_i, E, \hat{\alpha}_i, \hat{X}_{i0}). \tag{11}$$

The state $\hat{X}_i = \hat{X} \times (\{0, 1, \ldots, \lceil \frac{|a_1 a_2|}{T} \rceil, \infty\})^{j_i} \times \{H, F\}$ ($j_i \in \{1, \ldots, n_i\}$), where "$H$" means "healthy' and "$F$" means "faulty". For each $z_i \in Z_i$ and the corresponding $\hat{x}_i = ((x, |\hat{x}_i|_f), (|\hat{x}_i|_i^{j_i})^{j_i}, |\hat{x}_i|_d) \in \hat{X}_i$, denote $|\hat{x}_i|_f = |I_1(z_i)|_f$, $|\hat{x}_i|_i^{j_i} = |I_1(z_i)|_i^{j_i}$ and

$$|\hat{x}_i|_d = \begin{cases} F, & \text{if } z_i \in \mathcal{VC}_i \\ H, & \text{otherwise} \end{cases} \tag{12}$$

where we define the condition $z_i \in \mathcal{VC}_i$ as follows:

$$|I_1(z_i)|_f = K, |I_2(z_i)|_f = -1, |I_1(z_i)|_i^{j_i} > 0, |I_2(z_i)|_i^{j_i} > 0. \tag{13}$$

Clearly, $\hat{\mathcal{R}}_i$ can be seen as a fault automaton, in fact, each state $\hat{x}_i \in \hat{X}_i$ has a fault counting value, and $|\hat{x}_i|_d = F$ can be regarded as the fault states $\hat{x} \in \hat{X}$ satisfying $|\hat{x}|_f = K$ in the automaton $\hat{G}$ to determine the confusing state subset $Z^C_{E_k}$.

Example 5. (The reconstructed automaton): For the fault automaton $\hat{G}$ and the state subset $Z^C_{E_1}$ in Example 4, we run Algorithm 1 with $T=1$ to obtain $\mathcal{R}_1$ shown in Fig. 4(a). The crucial difference between $\mathcal{R}_1$ and $\hat{\mathcal{R}}_1$ [shown in Fig. 4(c)] is the third component $\{H, F\}$. For compactness, Fig. 4(b) shows the M-machine $\mathcal{M}_{E_1}(\mathcal{R}_1) = (Z_1, E \cup \{\epsilon\}, \delta_1, Z_{10})$ after omitting the states $z_1 \in Z_1$ satisfying $\mathcal{A}^{\mathcal{R}_1}(I_1(z_1)) \cap \{z'_1 \in Z_1 | |I_1(z'_1)|_f = K\} = \emptyset \ \vee \ |I_2(z_i)|_f > 0$ [according to (13), all states $\hat{x}_1 \in \hat{X}_1$ corresponding to the omitted states in Fig. 4(b) must satisfy $|\hat{x}_1|_d = H$]. For the string $fe_1e_2e_2 \in \mathcal{L}(\mathcal{M}_{E_1}(\mathcal{R}_1))$, we have $\delta_1(fe_1e_2e_2) = \{(((4,3),(1)^1),((2,-1),(0)^1)), (((4,3),(1)^1),((2,-1),(1)^1)), \ldots\}$, which corresponds to $\hat{\alpha}_1(fe_1e_2e_2) = \{((4,3),(1)^1,H),((4,3),(1)^1,F)\}$. The state $((4,3),(1)^1,F)$ corresponds to the fact, shown in Example 3, that the fault may not be diagnosed by a_1. □

Using the automaton $\hat{\mathcal{R}}_i$, we then construct the M-machine $\mathcal{M}_{E_k}(\hat{\mathcal{R}}_i)$ to obtain $Z^C_{E_k}$ and further run Algorithm 1 with $\hat{\mathcal{R}}_i$ and E_k to get the augmented delay recorder

$$\mathcal{R}_{i,k} = (X_{i,k}, E, \alpha_{i,k}, X_{i,k,0}) \tag{14}$$

where similar to a_i, we can obtain n_k and denote the delay value of a_k as $|\cdot|_k^{j_k}$ ($j_k \in \{1, \ldots, n_k\}$). Then, using E_k, the M-machine

$$\mathcal{M}_{E_k}(\mathcal{R}_{i,k}) = (Z_{i,k}, E \cup \{\epsilon\}, \delta_{i,k}, Z_{i,k,0}) \tag{15}$$

is obtained, where we still denote $I_1(z)$ as the first state component, $I_2(z)$ as the second state component for each state $z \in Z_{i,k}$. Finally, we define a diagnosis function $\psi : z \to \{H, F\}$ as follows: $\forall z \in Z_{i,k}$,

$$\psi(z) = \begin{cases} F, & \text{if } z \in \mathcal{VC}_k \\ H, & \text{otherwise} \end{cases} \tag{16}$$

where condition $z \in \mathcal{VC}_k$ is defined as follows:

$$|I_1(z)|_d = F, |I_2(z)|_f = -1, |I_1(z)|_k^{j_k} > 0, |I_2(z)|_k^{j_k} > 0. \tag{17}$$

With a slight abuse of notation, although the notation $|\cdot|_d$ is defined for states in $\hat{\mathcal{R}}_i$, we denote $|I_1(z)|_d = F$ for $I_1(z) \in X_{i,k}$ in $\mathcal{R}_{i,k}$. This is possible because $\mathcal{R}_{i,k}$ is built from $\hat{\mathcal{R}}_i$, the only difference being that $X_{i,k}$ contains the delay information of a_k. Similarly, we also allow the states in $X_{i,k}$ to use $|\cdot|_f$.

Now we are in the position to verify K^T-codiagnosability with the following theorem.

Theorem 1: Let G in (1) be the system model, E_1 and E_2 be the set of observable events for agents a_1 and a_2, f be the fault events, $\mathcal{M}_{E_k}(\mathcal{R}_{i,k})$ in (15) be the M-machine built from the augmented delay recorder $\mathcal{R}_{i,k}$ in (14). Then, $\mathcal{L}(G)$ is K^T-codiagnosable w.r.t. f iff

$$\forall z \in Z_{i,k}, \psi(z) = H. \tag{18}$$

Proof: ($\Rightarrow$) By contradiction, let us first suppose that $\mathcal{L}(G)$ is K^T-codiagnosable w.r.t. f while $\exists z \in Z_{i,k}$, s.t. $\psi(z) = F$. Then, we have that z satisfies (17).

First, we consider a_k, combining (17) and (13), we have $\exists j \in \{1, \ldots, n_k\}$, s.t. $|I_1(z)|_k^j > 0$, $|I_2(z)|_k^j > 0$, $|I_1(z)|_f = K$, $|I_2(z)|_f = -1$ and $I_1(z), I_2(z) \in X_{i,k}$.

Since $|I_1(z)|_k^j > 0$ and $|I_2(z)|_k^j > 0$, there must be a pair of event strings, $s^f, s^c_k \in \mathcal{L}(\mathcal{R}_{i,k}) : I_1(z) \in \alpha_{i,k}(s^f) \wedge I_2(z) \in \alpha_{i,k}(s^c_k) \wedge P_{E_k}(s^f) = P_{E_k}(s^c_k)$ such that two transitions are marked, where we denote the events of the marked transition as e^{f1} in s^f and e^{c1} in s^c_k. Then, we further denote $s^f = s^{f1}e^{f1}s^{f2}$ and $s^c_k = s^{c1}e^{c1}s^{c2}$. Recalling the FORWARD1 procedure in Algorithm 1, we have $P_{E_o}(s^{f1}e^{f1}) = P_{E_o}(s^{c1}e^{c1})$. We now consider the following two cases.

1) If the delay of an event e^{f2} (or e^{c2}) $\in E_o \backslash E_k$ in s^{f2} (s^{c2}) is recorded, then the system will enter $I_1(z)$ (or $I_2(z)$) within $\lceil \frac{|a_1 a_2|}{T} \rceil - 1$ steps after the occurrence of e^{f2} (or e^{c2}), that is, $0 < |I_1(z)|_k^j \leq \lceil \frac{|a_1 a_2|}{T} \rceil$ (or $0 < |I_1(z)|_k^j \leq \lceil \frac{|a_1 a_2|}{T} \rceil$). In this case, we further denote $s^{f2} = s^{f3}e^{f2}s^{f4}$ (or $s^{c2} = s^{c3}e^{c2}s^{c4}$), where $|s^{f4}| < \lceil \frac{|a_1 a_2|}{T} \rceil$ (or $|s^{c4}| < \lceil \frac{|a_1 a_2|}{T} \rceil$).

2) If no delay in s^f (or s_k^c) is recorded, then the system will enter $I_1(z)$ (or $I_2(z)$) without any occurrence of the events in $E_o \backslash E_k$, that is, $|I_1(z)|_k^j = \infty$ (or $|I_1(z)|_k^j = \infty$).

Recalling the FORWARD1 procedure in Algorithm 1, we know that $P_{E_o \backslash E_k}(s^{f1}e^{f1}) = P_{E_o \backslash E_k}(s^{f1}e^{f1}s^{f3})$ and $P_{E_o \backslash E_k}(s^{c1}e^{c1}) = P_{E_o \backslash E_k}(s^{c1}e^{c1}s^{c3})$ in case 1), and $P_{E_o \backslash E_k}(s^{f1}e^{f1}) = P_{E_o \backslash E_k}(s^f)$ and $P_{E_o \backslash E_k}(s^{c1}e^{c1}) = P_{E_o \backslash E_k}(s_k^c)$ in case 2). Then, we have $s^{f1}e^{f1} \in \zeta_{E_o \backslash E_k}^{\lceil \frac{|a_1 a_2|}{T} \rceil}(s^f)$ and $s^{c1}e^{c1} \in \zeta_{E_o \backslash E_k}^{\lceil \frac{|a_1 a_2|}{T} \rceil}(s_k^c)$, indicating that $P_{E_o}(\zeta_{E_o \backslash E_k}^{\lceil \frac{|a_1 a_2|}{T} \rceil}(s^f)) \cap P_{E_o}(\zeta_{E_o \backslash E_k}^{\lceil \frac{|a_1 a_2|}{T} \rceil}(s_k^c)) \neq \emptyset$.

Now we consider a_i: we know that $\mathcal{R}_{i,k}$ is built with Algorithm 1 from $\hat{\mathcal{R}}_i$ which is reconstructed from the M-machine $\mathcal{M}_{E_i}(\mathcal{R}_i) = (Z_i, E \cup \{\epsilon\}, \delta_i, Z_{i0})$. From $|I_1(z)|_d = F$, we have the corresponding $|\hat{x}_i|_d = F$, and further z_i satisfies $\exists j' \in \{1, \ldots, n_i\}$, s.t. $|I_1(z_i)|_i^{j'} > 0$, $|I_2(z_i)|_i^{j'} > 0$, $|I_1(z_i)|_f = K$ and $|I_2(z_i)|_f = -1$, where $I_1(z_i), I_2(z_i) \in X_i$ in $\mathcal{R}_i = (X_i, E, \alpha_i, X_{i0})$. Then, from $I_1(z) \in \alpha_{i,k}(s^f)$, we know that $I_1(z_i) \in \alpha_i(s^f)$. Recalling the property of M-machine, $\exists s_i^c \in \mathcal{L}(\mathcal{R}_i) : I_2(z_i) \in \alpha_i(s_i^c)$, s.t. $P_{E_i}(s_i^c) = P_{E_i}(s^f)$. And with a similar analysis as above, we have $P_{E_o}(\zeta_{E_o \backslash E_i}^{\lceil \frac{|a_1 a_2|}{T} \rceil}(s^f)) \cap P_{E_o}(\zeta_{E_o \backslash E_i}^{\lceil \frac{|a_1 a_2|}{T} \rceil}(s_i^c)) \neq \emptyset$.

To sum up, for the event string $s^f \in \mathcal{L}(\mathcal{R}_{i,k}) = \mathcal{L}(\mathcal{R}_i) = \mathcal{L}(G)$: $\Delta(s^f) = K, \forall l \in \{1, 2\}, P_{E_l}(s^f) = P_{E_l}(s_l^c), P_{E_o}(\zeta_{E_o \backslash E_l}^{\lceil \frac{|a_1 a_2|}{T} \rceil}(s^f)) \cap P_{E_o}(\zeta_{E_o \backslash E_l}^{\lceil \frac{|a_1 a_2|}{T} \rceil}(s_l^c)) \neq \emptyset$ and $\Delta(s_l^c) = -1$. In other words, $\mathcal{L}(G)$ is not K^T-codiagnosable w.r.t. f, resulting in a violation.

($\Leftarrow$) By contradiction, let us suppose that $\forall z \in Z_{i,k}, \psi(z) = H$ while $\mathcal{L}(G)$ is not K^T-codiagnosable w.r.t. f. Then, we have that there exists $s \in \mathcal{L}(G) : \Delta(s) = K$, such that condition (7) is violated for a_1 and a_2.

First, we consider $a_i : \exists s_i^c \in \mathcal{L}(\hat{G}) = \mathcal{L}(\mathcal{R}_i) : \Delta(s_i^c) = -1$, s.t. $P_{E_i}(s) = P_{E_i}(s_i^c)$, $P_{E_o}(\zeta_{E_o \backslash E_i}^{\lceil \frac{|a_1 a_2|}{T} \rceil}(s)) \cap P_{E_o}(\zeta_{E_o \backslash E_i}^{\lceil \frac{|a_1 a_2|}{T} \rceil}(s_i^c)) \neq \emptyset$.

Then, we have that $\exists e^{f1}, e^{c1} \in E_o \backslash E_i : s = s^{f1}e^{f1}s^{f2}, s_i^c = s^{c1}e^{c1}s^{c2}$, s.t. $P_{E_o}(s^{f1}e^{f1}) = P_{E_o}(s^{c1}e^{c1}) \in P_{E_o}(\zeta_{E_o \backslash E_i}^{\lceil \frac{|a_1 a_2|}{T} \rceil}(s)) \cap P_{E_o}(\zeta_{E_o \backslash E_i}^{\lceil \frac{|a_1 a_2|}{T} \rceil}(s_i^c))$. Since $P_{E_o}(s^{f1}e^{f1}) = P_{E_o}(s^{c1}e^{c1})$, the transitions of e^{f1} and e^{c1} in s and s_i^c must be marked by an index in line 2 of Algorithm 1. Nevertheless, after the first check in lines 3–9, the mark on e^{f1} and e^{c1} may be canceled, but an event in s^{f2} and an event in s^{c2} will still be marked according to the condition in line 5. Hence, we can regard e^{f1} and e^{c1} as the marked transition without loss of generality. Next, after the second check in lines 11–20, the transition e^{f1} and e^{c1} in s and s_i^c may not be marked, but there must be another pair of event strings marking e^{f1} and e^{c1} according to the conditions in lines 12 and 14. Hence, we can regard s and s_i^c as the pair of event strings where e^{f1} and e^{c1} are marked without loss of generality. Now we consider $j_i \in \{1, \ldots, n_i\}$ as the index that marks the transitions of e^{f1} and e^{c1} in s and s_i^c. Recalling the RECORD procedure in Algorithm 1, there are two cases to be analyzed as follows.

1) If the delay of an event e^{f2} (or e^{c2}) after e^{f1} (or e^{c1}) is recorded, then, we can denote $s^{f2} = s^{f3}e^{f2}s^{f4}$ (or $s^{c2} = s^{c3}e^{c2}s^{c4}$). Here, we know $|s^{f4}| < \lceil \frac{|a_1 a_2|}{T} \rceil$ (or $|s^{c4}| < \lceil \frac{|a_1 a_2|}{T} \rceil$), otherwise there will be a violation that $P_{E_o}(s^{f1}e^{f1}) \notin P_{E_o}(\zeta_{E_o \backslash E_i}^{\lceil \frac{|a_1 a_2|}{T} \rceil}(s))$ [or $P_{E_o}(s^{c1}e^{c1}) \notin P_{E_o}(\zeta_{E_o \backslash E_i}^{\lceil \frac{|a_1 a_2|}{T} \rceil}(s_i^c))$]. Since the delay is recorded as $\lceil \frac{|a_1 a_2|}{T} \rceil$ steps, there must be a state $x_i \in \alpha_i(s) : |x_i|_i^{j_i} > 0$ [or $x_i' \in \alpha_i(s_i^c) : |x_i'|_i^{j_i} > 0$].
2) If no delay after e^{f1} (or e^{c1}) is recorded, then there must be a state $x_i \in \alpha_i(s) : |x_i|_i^{j_i} = \infty$ (or $x_i' \in \alpha_i(s_i^c) : |x_i'|_i^{j_i} = \infty$).

Since $P_{E_i}(s) = P_{E_i}(s_i^c)$, we have $(x_i, x_i') \in \delta_i(s) \subseteq Z_i$ in $\mathcal{M}_{E_i}(\mathcal{R}_i)$ with $|x_i|_i^{j_i} > 0$, $|x_i'|_i^{j_i} > 0$, $|x_i|_f = K$ and $|x_i'|_f = -1$, which means the corresponding state $\hat{x}_i \in \hat{\alpha}_i(s)$ in $\hat{\mathcal{R}}_i$, as well as $x_{i,k} \in \hat{\alpha}_{i,k}(s)$ in $\mathcal{R}_{i,k}$, satisfies $|\hat{x}_i|_d = |x_{i,k}|_d = F$.

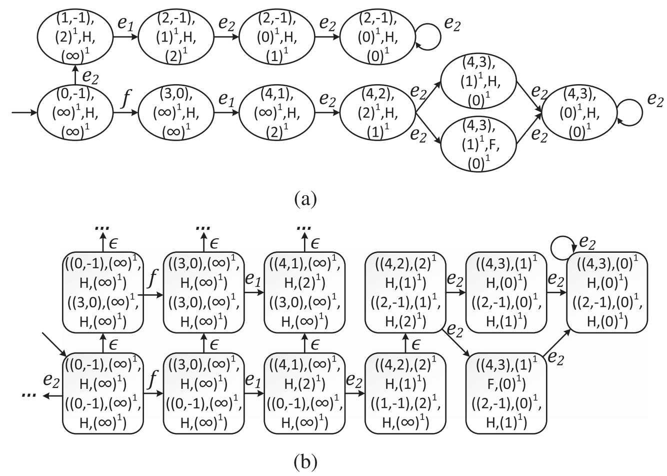

Fig. 5. Augmented delay recorder $\mathcal{R}_{1,2}$ and part of M-machine $\mathcal{M}_{E_2}(\mathcal{R}_{1,2})$ that may include the fault state. (a) $\mathcal{R}_{1,2}$. (b) Part of $\mathcal{M}_{E_2}(\mathcal{R}_{1,2})$.

Now we consider $a_k : \exists s_k^c \in \mathcal{L}(\hat{G}) = \mathcal{L}(\mathcal{R}_{i,k}) : \Delta(s_k^c) = -1$, s.t. $P_{E_k}(s) = P_{E_k}(s_k^c)$, $P_{E_o}(\zeta_{E_o \backslash E_k}^{\lceil \frac{|a_1 a_2|}{T} \rceil}(s)) \cap P_{E_o}(\zeta_{E_o \backslash E_k}^{\lceil \frac{|a_1 a_2|}{T} \rceil}(s_k^c)) \neq \emptyset$. Then, as well, there must be an index $j_k \in \{1, \ldots, n_k\}$ marking the relevant transitions in s and s_k^c. With a similar analysis as above, we have that $\exists (x_{i,k}, x_{i,k}') \in \delta_{i,k}(s) \subseteq Z_{i,k}$ in $\mathcal{M}_{E_k}(\mathcal{R}_{i,k})$, s.t. $x_{i,k} \in \alpha_{i,k}(s), x_{i,k}' \in \alpha_{i,k}(s_k^c), |x_{i,k}|_k^{j_k} > 0, |x_{i,k}'|_k^{j_k} > 0$. Since $|x_{i,k}|_d = F$ and $|x_{i,k}'|_f = -1$, we have $\psi((x_{i,k}, x_{i,k}')) = F$, resulting in a violation, which completes the proof. ∎

Example 6. (Verifying K^T-codiagnosability): Considering the automaton $\hat{\mathcal{R}}_1$ in Example 5, we build the M-machine $\mathcal{M}_{E_2}(\hat{\mathcal{R}}_1)$ to get $Z_{E_2}^C = \{(((4,3),(1)^1,F),\ ((2,-1),(0)^1,H))\}$. Then, we run Algorithm 1 with $\hat{\mathcal{R}}_1$, $\lceil \frac{|a_1 a_2|}{T} \rceil$, $Z_{E_2}^C$, E_o and E_2 to obtain the augmented delay recorder $\mathcal{R}_{1,2}$ as shown in Fig. 5(a). Finally, the M-machine $\mathcal{M}_{E_2}(\mathcal{R}_{1,2}) = (Z_{1,2}, E \cup \{\epsilon\}, \delta_{1,2}, Z_0)$ is constructed with E_2: Fig. 5(b) shows 11 of the 31 states of $\mathcal{M}_{E_2}(\mathcal{R}_{1,2})$, where the omitted 20 states $z \in Z_{1,2}$ obviously satisfy $|I_1(z)|_d = H$ according to (17). The only state z satisfying $|I_1(z)|_d = F$ is $(((4,3),(1)^1,F,(0)^1),((2,-1),(0)^1,H,(1)^1))$, however, we have $|((4,3),(1)^1,F,(0)^1)|_2^1 = 0$, violating (17). Hence, for any $z \in Z_{1,2}$, we have $\psi(z) = H$, indicating that $\mathcal{L}(G)$ is K^T-codiagnosable w.r.t. f with $K = 3$ and $T = 1$. Despite a_1 failing to diagnose the fault (cf. Example 3), the diagnosis task can be fulfilled thanks to a_2, indicating that distributed diagnosability is possible if and only if at least one of the agents can diagnose the faults with the information received from other agents. □

V. Conclusion

In this article, a novel framework has been presented to solve the problem of distributed fault diagnosis in discrete event systems with delays arising from transmission impairments. A new notion of K^T-codiagnosability was proposed that extends the well-known K-codiagnosability to the distributed setting. Accordingly, a novel delay recorder structure and a new diagnosis function were proposed to verify K^T-codiagnosability. Future work could consider more complex diagnosis problems with transmission impairments, such as distributed dynamic sensor activation.

REFERENCES

[1] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," *IEEE Trans. Autom. Control*, vol. 40, no. 9, pp. 1555–1575, Sep. 1995.

[2] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Berlin, Germany: Springer, 2009.

[3] S. Lafortune, F. Lin, and C. N. Hadjicostis, "On the history of diagnosability and opacity in discrete event systems," *Annu. Rev. Control*, vol. 45, pp. 257–266, 2018.

[4] X. Yin, J. Chen, Z. Li, and S. Li, "Robust fault diagnosis of stochastic discrete event systems," *IEEE Trans. Autom. Control*, vol. 64, no. 10, pp. 4237–4244, Oct. 2019.

[5] C. Keroglou and C. N. Hadjicostis, "Verification of AA-diagnosability in probabilistic finite automata is PSPACE-hard," in *Proc. IEEE 58th Conf. Decis. Control*, 2019, pp. 6712–6717.

[6] W. Wang, C. Gong, and D. Wang, "Optimizing sensor activation in a language domain for fault diagnosis," *IEEE Trans. Autom. Control*, vol. 64, no. 2, pp. 743–750, Feb. 2019.

[7] Y. Hu, Z. Ma, and Z. Li, "Design of supervisors for active diagnosis in discrete event systems," *IEEE Trans. Autom. Control*, vol. 65, no. 12, pp. 5159–5172, Dec. 2020.

[8] X. Yin and S. Lafortune, "A general approach for optimizing dynamic sensor activation for discrete event systems," *Automatica*, vol. 105, pp. 376–383, 2019.

[9] A. White, A. Karimoddini, and R. Su, "Fault diagnosis of discrete event systems under unknown initial conditions," *IEEE Trans. Autom. Control*, vol. 64, no. 12, pp. 5246–5252, Dec. 2019.

[10] R. Debouk, S. Lafortune, and D. Teneketzis, "Coordinated decentralized protocols for failure diagnosis of discrete event systems," *Discrete Event Dyn. Syst.*, vol. 10, no. 1/2, pp. 33–86, 2000.

[11] X. Yin and S. Lafortune, "Codiagnosability and coobservability under dynamic observations: Transformation and verification," *Automatica*, vol. 61, pp. 241–252, 2015.

[12] G. S. Viana and J. C. Basilio, "Codiagnosability of discrete event systems revisited: A new necessary and sufficient condition and its applications," *Automatica*, vol. 101, pp. 354–364, 2019.

[13] C. E. Nunes, M. V. Moreira, M. V. Alves, L. K. Carvalho, and J. C. Basilio, "Codiagnosability of networked discrete event systems subject to communication delays and intermittent loss of observation," *Discrete Event Dyn. Syst.*, vol. 28, no. 2, pp. 215–246, 2018.

[14] G. Viana, M. V. S. Alves, and J. C. Basilio, "Codiagnosability of networked discrete event systems with timing structure," *IEEE Trans. Autom. Control*, vol. 67, no. 8, pp. 3933–3948, Aug. 2022.

[15] C. Keroglou and C. N. Hadjicostis, "Distributed fault diagnosis in discrete event systems via set intersection refinements," *IEEE Trans. Autom. Control*, vol. 63, no. 10, pp. 3601–3607, Oct. 2018.

[16] M. Z. Veras, F. G. Cabral, and M. V. Moreira, "Distributed synchronous diagnosis of discrete event systems modeled as automata," *Control Eng. Pract.*, vol. 115, 2021, Art. no. 104892.

[17] K. Rudie and J. C. Willems, "The computational complexity of decentralized discrete-event control problems," *IEEE Trans. Autom. Control*, vol. 40, no. 7, pp. 1313–1319, Jul. 1995.

[18] X. Yin and S. Lafortune, "Minimization of sensor activation in decentralized discrete-event systems," *IEEE Trans. Autom. Control*, vol. 63, no. 11, pp. 3705–3718, Nov. 2018.

[19] F. Cassez and S. Tripakis, "Fault diagnosis with static and dynamic observers," *Fundam. Informaticae*, vol. 88, no. 4, pp. 497–540, 2008.

[20] F. Cassez, "The complexity of codiagnosability for discrete event and timed systems," *IEEE Trans. Autom. Control*, vol. 57, no. 7, pp. 1752–1764, Jul. 2012.

[21] L. B. Clavijo and J. C. Basilio, "Empirical studies in the size of diagnosers and verifiers for diagnosability analysis," *Discrete Event Dyn. Syst.*, vol. 27, pp. 701–739, 2017.