

Distributed Diagnosis of Place-Bordered Petri Nets

Sahika Genc, Stéphane Lafortune,

Department of Electrical Engineering and Computer Science, University of Michigan, 1301 Beal Avenue, Ann Arbor, MI 48109-2122, USA `stephane,sgenc@eecs.umich.edu`

Abstract—This paper studies on-line fault detection and isolation of modular dynamic systems modeled as sets of place-bordered Petri nets. The common places among the set of Petri nets modeling a system capture coupling of various system components. The transitions are labeled by events, some of which are unobservable, i.e., not directly recorded by the sensors attached to the system. The events whose occurrence must be diagnosed have unobservable transition labels. These events model faults or other significant changes in the system state. The existing theory of diagnosis of discrete-event systems is extended in the context of the above model. The modular structure of the system is exploited by a distributed algorithm for fault diagnosis. A Petri net diagnoser is associated to every Petri net and the diagnosers communicate in real-time during the diagnostic process when the token count of common places changes. A merge function is defined to combine the individual diagnoser states and recover the complete diagnoser state that would be obtained under a monolithic approach. Strategies that reduce the communication overhead are presented. The software implementation of the distributed algorithm is discussed.

Note to practitioner - In the last decade monitoring, fault detection, and diagnosis methodologies based on the use of discrete-event models have been successfully used in a variety of technological systems ranging from document processing systems to intelligent transportation systems. This paper was motivated by the problem of fault diagnosis for modular (distributed) dynamic discrete-event systems (DES). As a DES modeling formalism, Petri nets offer potential advantages in terms of the distributed representation of the system and of the ability to represent coupling of the system components. The systems studied in this paper are sets of modules coupled with each other through various system components and modeled using Petri nets. We present a distributed fault diagnosis algorithm which allows each module in the distributed system to diagnose its faults independently unless completion of a task requires the use of coupled components. In the case of coupling, modules communicate with each other to accurately diagnose the fault. The distributed

fault diagnosis algorithm recovers the monolithic diagnosis information at the cost of communication and growing communication overhead. To mitigate that problem, we present an improved version of the algorithm that significantly reduces the communication overhead. Finally, we introduce the software toolbox (written in MATLAB and integrated with AT&T Graphviz) and we present a case study of an example of a *Heating, Ventilation and Air-Conditioning System* where we use the software tool for modeling and analysis of the system.

Index Terms—Fault diagnosis, distributed algorithms, Petri nets, software implementation.

I. INTRODUCTION

This paper addresses the problem of detecting and isolating faults or other significant events in the behavior of a modular dynamic system that is modeled as a set of interacting Petri net modules. The events to be diagnosed, referred to as “faults” hereafter, are modeled as unobservable events in the respective system modules. Events are unobservable when they are not directly recorded by the sensors attached to the system. The common places among the set of Petri nets modeling a system capture coupling of various system components. The objective is to diagnose the occurrence of fault events based on the sequence of observed events and on the structure of the respective Petri net modules and their coupling by common places. It is sought to obtain a distributed diagnosis algorithm that takes advantage of the modular structure of the system.

The problem of fault diagnosis for discrete-event systems has received considerable attention in the last decade and diagnosis methodologies based on the use of discrete-event models have been successfully used in a variety of technological systems ranging from document processing systems to intelligent transportation systems; see [1] and the references therein. The methodology termed the “Diagnoser Approach”, introduced in [2] and subsequently extended in several works including [3], [4], is of particular relevance to the present paper. The key feature of the Diagnoser Approach is the use of a special discrete-event process called the *diagnoser*. The diagnoser is built from the system model and is used to

This research is supported in part by NSF grants ECS-0080406, CCR-0082784 and CCR-0325571, by ONR grant N00014-03-1-0232, and by grant from the Xerox University Affairs Committee. The first author wishes to acknowledge support from a Barbour Fellowship from the Horace H. Rackham School of Graduate Studies at the University of Michigan.

(i) test the diagnosability properties of the system and (ii) perform on-line monitoring of the system for the purpose of fault diagnosis. The above references regarding the Diagnoser Approach are all based on the use of automata models for the system under consideration, leading to the construction of automata diagnosers.

This paper is concerned with discrete-event systems that are modeled by Petri nets. The use of Petri nets instead of automata offers potential advantages in system modeling and analysis, especially in terms of the distributed representation of the system state and of the ability to represent coupling of system components by means of common places.

Petri net models have been employed to solve problems of state observability, system monitoring, alarm analysis, and fault diagnosis in several works, including [5], [6], [7], [8], [9], [10], [11]. However, to the best of our knowledge, our DDC-2 and DDC- M algorithms are the first to explore the extension of the Diagnoser Approach of [2] to modular discrete-event systems modeled by Petri nets.

Systems possessing modular structures are receiving more and more attention in the recent literature on diagnosis, verification, and control of discrete-event systems; see, e.g., [4], [8], [9], [12], [13]. The suitability of Petri nets to model distributed systems was a key motivation for the use of Petri net structures in the work in [8] on alarm supervision in telecommunication networks. The same consideration motivates our choice of Petri net structures as a means to mitigate the combinatorial explosion that occurs when modular models are converted to monolithic ones. Our approach is different from that in related work such as [4], [8], [13], [14] and thus our work is complementary to these references.

Our investigations on the problem of fault diagnosis of Petri nets were first reported in [15] where the notion of *centralized* (monolithic) Petri net diagnosers is introduced. Petri net diagnosers serve the same purpose as the automata diagnosers in [2] for on-line monitoring and diagnosis of a system, but they are based on the same Petri net structure as the system model, unlike diagnoser automata which require a conversion of the system model from nondeterministic to deterministic. Our initial work reported in [15] also considered systems composed of *two* Petri nets sharing a set of common places, leading to a distributed diagnosis algorithm with communication abbreviated as “DDC-2” hereafter.¹ In this paper, we consider the case of modular systems consisting of a *set* of M place-bordered Petri nets. We present two new

algorithms, one termed DDC- M , that extends DDC-2 to the case of multiple modules, and the other termed DDC- M with *fixed-size message labels* which uses an encoding of messages and significantly improves upon the real-time communication requirements. A preliminary version of DDC- M , without message encoding, is presented without a correctness proof in [16]. Clearly, the monolithic approach is a special case of the modular approach where the set of place-bordered Petri nets is a singleton.

Our objectives in the case of the modular approach are: (i) to perform on-line diagnosis of faults in each module and (ii) to recover the monolithic diagnosis information obtained when all the modules in the system are combined into a single module that preserves the behavior of the underlying modular system. The first objective requires a Petri net diagnoser to be attached to each module in the system. Each Petri net diagnoser has local information on the structure of the module, and observes and diagnoses the fault types of the module it is attached to. The diagnoser has shared information on its places that are coupled with other modules in the system. The second objective requires the Petri net diagnosers to communicate among each other. Each communicating Petri net diagnoser sends messages to the diagnosers it is coupled with when a change occurs in the shared information (i.e., a change in the token count of common places) upon observation of an event. The communication of messages triggers the other diagnosers to update their diagnosis information based on the change in the shared information. The communication and update of the diagnosis information are the two key features that allow the modular diagnosis approach to correctly recover the monolithic diagnosis information. In general, a modular approach that does not consider the coupling of modules through shared information incorrectly estimates the monolithic diagnosis information. We present in Figure 1 the general architecture of the modular diagnosis approach described so far.

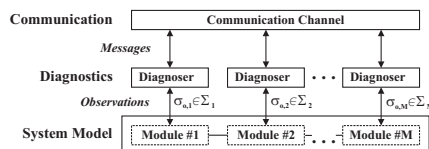


Fig. 1. General architecture of modular diagnosis approach.

The remainder of this paper is organized as follows. In Section II, we start with a brief summary of terms used throughout the paper. In Section III, we state the problem of fault diagnosis. The distributed diagnosis algorithm is based on communicating Petri net diagnosers.

¹DDC-2 is denoted by DDC in [15]; the “-2” label has been added in this paper for the sake of clarity.

The structure and dynamics of communicating Petri net diagnosers are defined in Section IV. In Section V, we present the first version of our distributed algorithm with communication for diagnosing systems composed of M modules, DDC- M where $M \geq 2$. For the sake of clarity of presentation, this initial version does not use encoding of messages. In Section VII, we state results about the correctness of the DDC- M . In Section VIII, we present the DDC- M with fixed-size message labels. In Section IX, we study an example of an *Heating, Ventilation and Air-Conditioning System*, which consists of a valve, pump and load module. Finally, in Section X, we give some concluding remarks. We give the proofs of the results about the correctness of the DDC- M in Appendix.

II. PRELIMINARIES

We start with some definitions (stated briefly since they are standard; see, e.g., Chapter 4 of [17] for further details). A Petri net graph is defined as $\mathcal{N} = \langle P, T, A, w \rangle$, where P and T are finite sets of places and transitions, respectively, A is the set of arcs from places to transitions and from transitions to places, and $w : A \rightarrow \mathbb{Z}^+$ is the weight function on the arcs. We denote by $W(P, t)$ the row vector of size equal to the number of places in P and whose i^{th} column is equal to $w(t, p_i) - w(p_i, t)$ where $p_i \in P$ and $t \in T$.

A labeled Petri net is defined as $(\mathcal{N}, \Sigma, l, x_0)$, where Σ is the set of events, $l : T \rightarrow \Sigma$ is the transition labeling function, and x_0 is the initial state. A transition $t \in T$ can fire from $x \in X$, where X is the state space of the labeled Petri net, if and only if t is feasible (enabled) from x . When t fires, the state transition function $f : X \times T \rightarrow X$ gives the resulting state according to the usual Petri net dynamics.

Some of the events in Σ are observable, i.e., their occurrence can be observed (detected by sensors), while the other events are unobservable; thus $\Sigma = \Sigma_o \cup \Sigma_{uo}$. The set of fault events Σ_f is a subset of Σ_{uo} . We partition the set of faults into disjoint sets where each set corresponds to a different fault type. This is because it might not be necessary to detect and isolate uniquely every fault event, but only the occurrence of one among a subset (type) of fault events. We denote by Σ_{Fk} the set of fault events corresponding to a type k fault.

III. PROBLEM STATEMENT

As was mentioned earlier in the introduction, the system to be diagnosed is modeled as a collection of Petri nets (modules) coupled with each other through common places. The choice of Petri nets to model a system with

a modular structure is a natural one. Examples of Petri nets coupled by means of common places, hereafter called *place-bordered Petri nets*, are found in many industrial applications such as automated manufacturing and communication systems; see, e.g., [18], [19], [20], [21].

Formally, the system to be diagnosed is the set \mathcal{S} of place-bordered Petri nets defined as

$$\mathcal{S} = \{(\mathcal{M}_m, \mathcal{P}_m) : m = 1, 2, \dots, M\} \quad (1)$$

where

$$\mathcal{M}_m = (\mathcal{N}_m, \Sigma_m, l_m, x_0^m), \quad (2)$$

is a labeled Petri net and

$$\mathcal{P}_m = \{P_{m,i} \subseteq P_m : i = 1, 2, \dots, M \text{ and } i \neq m\} \quad (3)$$

is a set of subsets of P_m where each subset $P_{m,i}$ is the set of common places between module m , \mathcal{M}_m , and module i , \mathcal{M}_i . By definition, the transition sets of the \mathcal{N}_m Petri net graphs are mutually disjoint.

We assume that the place-bordered Petri nets in the system operate as a single entity. Intuitively speaking, there is a global clock which sets the order in which modules execute their observable events during the operation of the system. We present in Figure 2 a conceptual view of a system of six place-bordered nets. In the figure, we draw dashed lines between the modules and put the common places on these dashed lines to illustrate the fact that the modules are isolated from each other except for the common places. We present in Figure 3 the implementation of the modular approach on a system of six place-bordered Petri nets. In the figure, we illustrate with a box the communicating Petri net diagnoser attached to a module and with the arrows drawn between the diagnosers the communication channels linking the diagnosers that have common places.

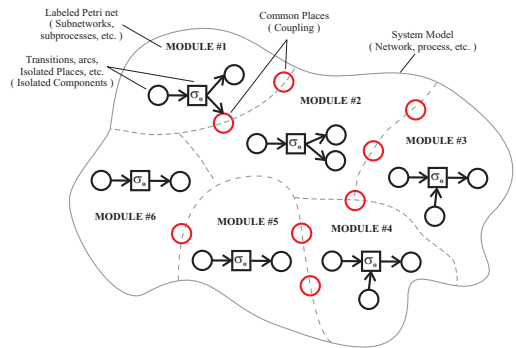


Fig. 2. System with six place-bordered nets.

The modular approach has a certain amount of *robustness* over the monolithic one, since each diagnoser in the modular approach has local knowledge of the monolithic

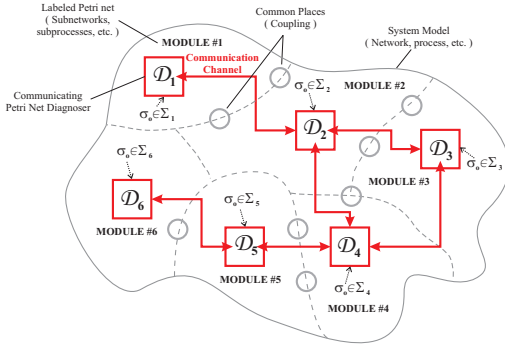


Fig. 3. System with six place-bordered nets.

system. The approach also has practical advantages in the sense that the modules are isolated from each other and do not share any structural information. When replacing one or several modules in the system, the rest of the modules in the system and the corresponding diagnosis devices stay the same as long as the information shared is not changed.

In the rest of the paper, we present in detail our modular diagnosis approach that achieves the objectives described in the introduction and restated in this section. We also define a method that implements a coding technique to reduce the size of the messages communicated while still recovering the monolithic diagnosis information.

IV. COMMUNICATING PETRI NET DIAGNOSERS

As was mentioned in Sections I and III, the communicating Petri net diagnosers introduced in [15] serve the same purpose as the automata diagnosers introduced in [2] for on-line diagnosis of faults or other significant events in behavior of the system. However, communicating Petri net diagnosers and automata diagnosers have different structures. A communicating Petri net diagnoser inherits the Petri net structure of the underlying system whereas an automaton diagnoser is obtained by an algorithm that incorporates the conversion of a nondeterministic automaton to a deterministic one. The diagnoser and the underlying net to be diagnosed have the same structure, but they do not have the same dynamics.

A communicating Petri net diagnoser, upon observation of an event, estimates the states the system could be in. Thus, a communicating Petri net diagnoser state contains a set of system states. The diagnoser state also carries diagnosis information, i.e., *fault label*, that provides information on the fault types that may have occurred. Moreover, a communicating Petri net diagnoser has a priori information on its common places with the other (neighbor) modules in the system. The diagnoser

memorizes the history of changes on the common places for each neighbor module and stores this history in the diagnoser state during the operation of the system. Since it is this history of changes that is communicated between the diagnosers, we call the corresponding part of the diagnoser state *message label*. Thus, in general, a communicating Petri net diagnoser state contains three parts: (i) a set of system states, (ii) fault label, and (iii) message labels for each neighbor module. In the case of a single module, the diagnoser state does not have the message label part since there is no other module to communicate with.

We now present the formal definitions of the structure and the dynamics of communicating Petri net diagnosers. We also restate the required knowledge on Petri net diagnosers to form a complete set of equations correctly describing communicating Petri net diagnosers.

In order to perform modular diagnosis we *assume* the following three conditions on the place-bordered Petri nets: (i) for each module $\mathcal{M}_m \in \mathcal{S}$, there exists another module $\mathcal{M}_n \in \mathcal{S}$ such that the set of common places between \mathcal{M}_m and \mathcal{M}_n , $P_{m,n}$, is not the empty set, (ii) $\forall \mathcal{M}_m \in \mathcal{S}, \exists \mathcal{M}_n \in \mathcal{S}, \Sigma_m \cap \Sigma_n = \emptyset$, (iii) $\forall \mathcal{M}_m \in \mathcal{S}, \forall t \in T_m$, if t puts tokens into or removes tokens from $P_{m,n}$ for some $\mathcal{M}_n \in \mathcal{S}$, then $l_m(t) \in \Sigma_{o,m}$. The motivation for labeling transitions putting tokens into or removing tokens from the common places with observable events is to allow communication between diagnosers to be triggered by observable events.

As was explained in Section III, we attach a communicating Petri net diagnoser to each module in the set \mathcal{S} of place-bordered Petri nets that form the system (see, e.g., Figure 3). We denote the diagnoser attached to module $(\mathcal{M}_m, \mathcal{P}_m)$ with the pair $(\mathcal{D}_m, \mathcal{P}_m)$ where $\mathcal{D}_m = (\mathcal{N}_m, \Sigma_m, l_m, x_0^{d,m}, \Delta_{f,m})$, $\Delta_{f,m}$ is the set of fault types of \mathcal{D}_m , and \mathcal{P}_m is as defined in Equation (3). The set of communicating Petri net diagnosers for the set of place-bordered Petri nets \mathcal{S} is denoted by $\mathcal{S}_{\mathcal{D}}$.

The type of communicating Petri net diagnosers we study in this paper were first defined in [15]. The communicating Petri net diagnosers in this paper differ from those in [15] in terms of the structure of message labels. We present the salient features of these diagnosers.

The diagnoser state x_d^m of module $\mathcal{D}_m \in \mathcal{S}_{\mathcal{D}}$ is a matrix of the form

$$\begin{pmatrix} - & - & - \\ x_s^m(i) & x_f^m(i) & x_l^m(i) \\ - & - & - \end{pmatrix} \quad (4)$$

where $x_s^m(i)$ denotes the state in row i of diagnoser state x_d^m , $x_f^m(i)$ denotes the corresponding fault label, and $x_l^m(i)$ denotes the corresponding message label. The state part

$x_s^m(i)$ of each row i corresponds to one possible state of \mathcal{M}_m following the occurrence of the observed sequence of events.

The diagnoser state transition function of $\mathcal{D}_m \in \mathcal{S}$ is of the form $f_{d,m} : X_d^m \times \Sigma_{o,m} \rightarrow X_d^m$, where X_d^m is the state space of \mathcal{D}_m . Given the diagnoser state $x_d^m \in X_d^m$ and the observable event $a \in \Sigma_{o,m}$, then $f_{d,m}(x_d^m, a)$ is defined only if there exists some $t \in T_m$ labeled with the observable event a and enabled from the state part of some row i of x_d^m . In that case, $f_{d,m}(x_d^m, a)$ is the listing of elements in the set

$$\cup_{u \in S_m(x_d^m, a)} UR_m(u), \quad (5)$$

where: (i) $S_m(x_d^m, a)$ is the set of states with the corresponding fault and message labels reached from the rows of x_d^m by firing transitions labeled with the observable event a in \mathcal{M}_m ; and (ii) $UR_m(u)$ is the set of states with the corresponding fault and message labels reached from u by firing the enabled transitions labeled with unobservable events. Let there be I rows in x_d^m . Formally, we have

$$\begin{aligned} S_m(x_d^m, a) &= \cup_{1 \leq i \leq I} \cup_{t \in B_m(x_d^m(i), a)} \\ &\{ (u_s^m | u_f^m | u_l^m) : u_s^m = f_m(x_s^m(i), t), u_f^m = x_f^m(i), \\ &\forall \mathcal{M}_n \in \mathcal{S} \setminus \mathcal{M}_m \text{ such that } P_{m,n} \neq \emptyset, \\ &u_l^m(P_{m,n}) = [x_l^m(i, P_{m,n}) W(P_{m,n}, t)] \}, \end{aligned} \quad (6)$$

where $B_m(x_d^m(i), a)$ is the set of $t \in T_m$ enabled from $x_d^m(i)$ and labeled with $a \in \Sigma_{o,m}$, and $W_{P_{m,n}}(t)$ is the weighting vector for t and the common places $P_{m,n}$ of \mathcal{M}_m and \mathcal{M}_n .

We define the unobservable reach for each $u \in S_m(x_d^m, a)$ as

$$\begin{aligned} UR_m(u) &= \{ (y_s | y_f | y_l) : \exists \bar{t} \in T_m^*, l_m(\bar{t}) \in \Sigma_{uo,m}^*, \\ &(y_s = f_m(u_s, \bar{t})), \forall k \in \Delta_{f,m} \\ &\left(y_f(k) = \begin{cases} 1, & \text{if } l(\bar{t}) \text{ contains an event in } \Sigma_{Fk}, \\ u_f(k), & \text{otherwise,} \end{cases} \right), \\ &\text{and } (y_l = u_l) \}. \end{aligned} \quad (7)$$

Fault labels are used as in automata diagnosers to memorize the occurrence of a fault event in the diagnoser state. Overall, in the fault label of a diagnoser state, each column corresponds to a fault type. Examination of a given column of the fault label in a diagnoser state reveals the current status of the diagnosis of the corresponding fault type (say Fk): (i) all rows have label 0 implies that a fault of Type Fk did not occur; (ii) some rows have label 0 and some rows have label 1 implies that a fault of Type Fk possibly occurred (“ Fk -uncertain state” in the terminology of [2]); (iii) all rows have label 1 implies that a fault of Type Fk occurred for sure (“ Fk -certain state” in the terminology of [2]).

The definition of message label is embedded in Equations (6) and (7). This is because the message label is based on the state evolution of the labeled Petri net and is formed using the structure of the Petri net graph. For convenience, we divide the message label into different parts where each part pertains to common places (if any) between two given modules.

We now present an example to illustrate the main notions and notation introduced in this section.

Example 1: Suppose that \mathcal{M}_m and \mathcal{M}_n are two coupled modules in \mathcal{S} . The diagnoser state x_d^m for \mathcal{D}_m is of the following form

$$x_d^m = \left(\begin{array}{c|c|c|c} a_1 & h_1 & \alpha_1 & \gamma_1 \\ \hline a_2 & h_2 & \alpha_2 & \gamma_2 \\ \hline x_s^m & x_f^m & x_l^m(P_{m,n}) & \end{array} \right), \quad (8)$$

where α_i for $i = 1, 2$ denotes the message label between the modules \mathcal{D}_m and \mathcal{D}_n , γ_i for $i = 1, 2$ denotes the message label for all modules $\mathcal{M}_{n'} \in \mathcal{S}$ that are coupled with \mathcal{M}_m and $n' \neq n$.

Suppose that the event $\sigma_o \in \Sigma_{o,m}$ is observed and the next diagnoser state of \mathcal{D}_m is $y_d^m = f_{d,m}(x_d^m, \sigma_o)$. Let t_1 and t_2 be enabled from the first and second row of x_d^m , respectively, and $l_m(t_1) = l_m(t_2) = \sigma_o$, i.e., $t_1, t_2 \in B_m(x_d^m(i), \sigma_o)$. Let $w_i = W(P_m, t_i)$ and $w_i(P_{m,n}) = W(P_{m,n}, t_i)$ for all $i = 1, 2$. In words, w_i denotes the difference between the number of tokens put into and removed from the places of \mathcal{M}_m when t_i is fired from a_i , and $w_i(P_{m,n})$ denotes the part of w_i that corresponds to the common places between \mathcal{M}_m and \mathcal{M}_n . Then, the set of states reached from a_i by firing transition t_i labeled with the observable event σ_o is formed by Equation (6) as follows

$$\begin{aligned} S_m(x_d^m, \sigma_o) &= \{ (a_1 + w_1 | h_1 | \alpha_1 w_1(P_{m,n}) : \gamma_1), \\ &(a_2 + w_2 | h_2 | \alpha_2 w_2(P_{m,n}) : \gamma_2) \}, \end{aligned}$$

where $\gamma_i'(P_{m,n'}) = [\gamma_i(P_{m,n'}) w_i(P_{m,n'})]$ for $i = 1, 2$ and for all modules $\mathcal{M}_{n'} \in \mathcal{S}$ coupled with \mathcal{M}_m except \mathcal{M}_n .

Suppose that there exists $\bar{t}_i \in T_m^*$ where $l(\bar{t}_i) \in \Sigma_{uo,m}^*$ such that \bar{t}_i is enabled from $a_i + w_i$ for $i = 1, 2$. Let $\bar{w}_i = W(P_m, \bar{t}_i)$ and $\bar{w}_i(P_{m,n}) = W(P_{m,n}, \bar{t}_i)$ for $i = 1, 2$. Then, the unobservable reach, defined by Equation (7), is

$$\begin{aligned} UR_m(S_m(x_d^m, \sigma_o)) &= \\ &\{ (a_1 + w_1 | h_1 | \alpha_1 w_1(P_{m,n}) : \gamma_1), \\ &(a_2 + w_2 | h_2 | \alpha_2 w_2(P_{m,n}) : \gamma_2), \\ &(a_1 + w_1 + \bar{w}_1 | h_1' | \alpha_1 w_1(P_{m,n}) : \gamma_1'), \\ &(a_2 + w_2 + \bar{w}_2 | h_2' | \alpha_2 w_2(P_{m,n}) : \gamma_2') \} \end{aligned} \quad (9)$$

where for all $k \in \Delta_{f,m}$ $h_i'(k) = 1$ if $l_m(\bar{t}_i)$ contains an event in Σ_{Fk} , otherwise $h_i'(k) = h_i(k)$ for $i = 1, 2$. The unobservable reach does not result in a change in message labels, since by assumption the transitions removing

tokens from or putting tokens into common places are labeled with observable events. As stated in Equation (5), the next diagnoser state $y_d^m = f_{d,m}(x_d^m, \sigma_o)$ is the listing of the elements of $UR_m(S_m(x_d^m, \sigma_o))$ in Equation (9). \square

The module and corresponding diagnoser have the same Petri net graph. Since the modules do not have disjoint sets of places, they can effect each other's states via the common (shared) places. If diagnosers are not informed of each others token additions/removals for the common places, then they incorrectly estimate the monolithic diagnoser state. Thus, they incorrectly estimate the fault information. As stated in the previous sections, we overcome this problem by defining a communication protocol between diagnosers.

In the following section, when we define the communication protocol, we will need the following notation for prefixes and suffixes of message labels. Suppose $y_d^m = f_{d,m}(x_d^m, a)$ for some $x_d^m \in X_d^m$ and $a \in \Sigma_{o,m}$. Then, for some $\mathcal{M}_n \in \mathcal{S}$ and rows i, j of x_d^m, y_d^m , respectively, if $y_l^m(j, P_{m,n}) = (x_l^m(i, P_{m,n}) \ W(P_{m,n}, t))$, then $y_l^m(j, P_{m,n}).Pfx = x_l^m(i, P_{m,n})$ and $y_l^m(j, P_{m,n}).Sfx = W(P_{m,n}, t)$.

V. COMMUNICATION PROTOCOL

We now formalize our algorithm for Distributed Diagnosis with Communicating Petri net diagnosers for a system with M modules. At this point, we are presenting a version of DDC- M where messages grow each time an observable event forces a communication. The purpose of presenting this version of the DDC- M is to illustrate the key features of our approach to distributed diagnosis with communication. In Section VIII, we present a modified version of DDC- M with messages of fixed-size, which is much preferable for implementation purposes.

DDC- M is composed of Algorithms 1 and 2 which are presented below. Algorithm 1 pertains to diagnoser state updates and if necessary generation of messages upon occurrence of an observable event at one module. Algorithm 2 pertains to diagnoser state updates upon reception of a message from another module. Pseudocode descriptions of Algorithms 1 and 2 are given in the tables below. We provide some explanations for the different lines in these two algorithms.

Algorithm 1: Line 1 considers that an observable event σ_{or} has occurred. The module the event occurs at is identified in line 2 and called hereafter the *master* module. In line 3, the diagnoser state of the master module is updated for the observed event according to the diagnoser state transition function. Then, all other modules that have common places with the master module, referred to as the *neighbor* modules hereafter, need to be considered (line 4). For those neighbor modules

whose common places with the master module were affected (addition and/or removal of tokens) by the execution of the observable event, lines 6-12 need to be performed. (Recall the assumption that transitions into common places are labeled by observable events.) In lines 6-12, the appropriate message for the communication from the master module to the neighbor module is constructed. This message consist of the message labels of the relevant rows of the master's diagnoser state, namely the rows for which tokens were removed and/or added in common places. Note that each row of the message is composed of a prefix (previous message label) and a suffix (most recent update on common places). The resulting of a message on the diagnoser state of the neighbor module is captured by the function *UDSC* in line 13, which is evaluated by Algorithm 2.

Algorithm 2: The algorithm is triggered by the reception of a message by a given module, which will result in an update of the diagnoser state at that module. The new diagnoser state is initialized in line 1. Then, the algorithm loops over the rows of the prefix part of the message received (line 2) and over the rows of the current message label in the diagnoser state (line 3) in order to find matches (line 4). Each match triggers the construction of a new row for the module's updated diagnoser state (lines 5 to 9). The construction of this row involves using the suffix of the message received to update to state of the common places affected and leaving the states of the other places unchanged (line 5). The fault label of the new row is carried over from that of the row that triggered the match since the event involved in the transition is an observable event (line 6). The suffix of the message received is appended to the appropriate part of the message label of the new row (line 7) while the rest of the message label is carried over (lines 8 and 9). The complete row constructed as described is added to the updated diagnoser state (line 11). The listing of all rows constructed by the above process for all matches in line 4 is the value returned by the function *UDSC*. Note that it is not necessary to perform the unobservable reach since we assume that transitions removing tokens from the common places are labeled by observable events.

We present an illustrative example to better understand the steps of Algorithms 1 and 2.

Example 2: Suppose that \mathcal{M}_m and \mathcal{M}_n are two coupled modules in \mathcal{S} . The diagnoser states x_d^m and x_d^n of \mathcal{D}_m and \mathcal{D}_n , respectively, are given as follows:

$$x_d^m = \left(\begin{array}{c|c|c|c} a_1 & h_1 & \alpha_1 & \gamma_1 \\ a_2 & h_2 & \underbrace{\alpha_2}_{x_l^m(P_{m,n})} & \gamma_2 \end{array} \right), \quad (10)$$

where α_i for $i = 1, 2$ denotes the message label between

Algorithm 1 Distributed Diagnosis with Communication

```

1: Upon occurrence of an observable event  $\sigma_o$ 
2: Find  $\mathcal{M}_m$  such that  $\sigma_o \in \Sigma_m$ ,
3:  $x_{d,r}^m \leftarrow f_{d,m}(x_{d,r-1}^m, \sigma_o)$ ,
4: for all  $\mathcal{D}_n \in \mathcal{S}_{\mathcal{D}}$  such that  $P_{m,n} \neq \emptyset$  do
5:   if  $\{W(P_{m,n}, t) \in B_m(x_{d,r-1}^m, \sigma_o)\} \neq \{\emptyset\}$  then
6:      $Mesg_{m,n} \leftarrow \{ \}$ ,
7:     for all  $j=1$ : Number of rows of  $x_{d,r}^m$  do
8:        $Mesg_{m,n}.Pfx(j) \leftarrow x_{d,r}^m(j, P_{m,n}).Pfx$ ,
9:        $Mesg_{m,n}.Sfx(j) \leftarrow x_{d,r}^m(j, P_{m,n}).Sfx$ ,
10:       $Mesg_{m,n}(j) \leftarrow (Mesg_{m,n}.Pfx(j), Mesg_{m,n}.Sfx(j))$ ,
11:    end for
12:    Send all different rows of  $Mesg_{m,n}$ ,
13:     $x_{d,r}^n \leftarrow UDSC(x_{d,r-1}^n, Mesg_{m,n})$ ,
14:  end if
15: end for

```

Algorithm 2 Update of Diagnoser State upon Communication

```

Require:  $x_{d,r-1}^n, Mesg_{m,n}$ 
1:  $X_{d,r}^n \leftarrow \{ \}$ ,
2: for all  $i = 1$ : Number of rows of  $Mesg_{m,n}.Pfx$  do
3:   for all  $j = 1$ : Number of rows of  $x_{d,r-1}^n(P_{m,n})$  do
4:     if  $Mesg_{m,n}.Pfx(i) = x_{d,r-1}^n(j, P_{m,n})$  then
5:        $y_s(P_{m,n}) \leftarrow x_{d,r-1}^n(j, P_{m,n}) + Mesg_{m,n}.Sfx(i)$ ,
6:        $y_s(P_n \setminus P_{m,n}) \leftarrow x_{d,r-1}^n(j, P_n \setminus P_{m,n})$ ,
7:        $y_f \leftarrow x_{d,r-1}^n(j)$ ,
8:        $y_i(P_{m,n}) \leftarrow (x_{d,r-1}^n(j, P_{m,n}), Mesg_{m,n}.Sfx(i))$ 
9:       for all  $\mathcal{D}_q \in (\mathcal{S}_{\mathcal{D}} \setminus \mathcal{D}_m)$  such that  $P_{n,q} \neq \emptyset$  do
10:         $y_i(P_{n,q}) \leftarrow x_{d,r-1}^n(j, P_{m,n})$ 
11:      end for
12:       $X_{d,r}^n \leftarrow X_{d,r}^n \cup \{y_s | y_f | y_i\}$ 
13:    end if
14:  end for
15:  $UDSC(x_{d,r-1}^n, Mesg_{m,n}) \leftarrow$  Listing of the set  $X_{d,r}^n$ 

```

the modules \mathcal{D}_m and \mathcal{D}_n (i.e., $P_{m,n} \neq \emptyset$), and γ_i for $i = 1, 2$ denotes the message labels for all $\mathcal{D}_{n'} \in \mathcal{S}_{\mathcal{D}}$ that \mathcal{D}_m is coupled with except $\mathcal{D}_{n'}$;

$$x_d^n = \begin{pmatrix} b_1 & | & k_1 & | & \beta_1 & : & \delta_1 \\ b_2 & | & k_2 & | & \underbrace{\beta_2} & : & \delta_2 \\ & & & & x_i^n(P_{m,n}) & & \end{pmatrix}, \quad (11)$$

where β_i for $i = 1, 2$ denotes the message label between the modules \mathcal{D}_m and \mathcal{D}_n and, δ_i for $i = 1, 2$ denotes the message labels for all $\mathcal{D}_{m'} \in \mathcal{S}_{\mathcal{D}}$ that \mathcal{D}_n is coupled with except $\mathcal{D}_{m'}$.

Suppose that the event $\sigma_o \in \Sigma_{o,m}$ is observed, then the new diagnoser state $y_d^m = f_{d,m}(x_d^m, \sigma_o)$ of \mathcal{D}_m is constructed as shown in Example 1 and is in the form

$$y_d^m = \begin{pmatrix} a_1 + w_1 & | & h_1 & | & \alpha_1 w_1(P_{m,n}) & : & \gamma'_1 \\ a_2 + w_2 & | & h_2 & | & \alpha_2 w_2(P_{m,n}) & : & \gamma'_2 \\ a_1 + w_1 + \overline{w_1} & | & h'_1 & | & \alpha_1 w_1(P_{m,n}) & : & \gamma'_1 \\ a_2 + w_2 + \overline{w_2} & | & h'_2 & | & \alpha_2 w_2(P_{m,n}) & : & \gamma'_2 \end{pmatrix}. \quad (12)$$

Suppose that $w_i(P_{m,n})$ for $i = 1, 2$ are not vectors of zeros. That is, the occurrence of σ_o results in a change in the token distribution of the common places between the modules \mathcal{D}_m and \mathcal{D}_n . Then, the occurrence of σ_o triggers a communication between \mathcal{D}_m and \mathcal{D}_n .

Since by assumption $\sigma_o \in \Sigma_{o,m}$, \mathcal{D}_m is the master module. Then, upon occurrence of σ_o , \mathcal{D}_m sends a message to \mathcal{D}_n . The message is the message label of \mathcal{D}_m for \mathcal{D}_n . The message label, extracted from the diagnoser state y_d^m in Equation (12), is as follows:

$$y_i^m(P_{m,n}) = \begin{pmatrix} \alpha_1 & w_1(P_{m,n}) \\ \alpha_2 & w_2(P_{m,n}) \end{pmatrix}. \quad (13)$$

Suppose that $\beta_1 = \alpha_1$ and $\beta_2 = \alpha_2$. Upon reception of the message \mathcal{D}_n updates x_d^n to y_d^n based on the message from \mathcal{D}_m (as defined in Algorithm 2) as follows

$$y_d^n = \begin{pmatrix} b'_1 & | & k_1 & | & \beta_1 w_1(P_{m,n}) & : & \delta_1 \\ b'_2 & | & k_2 & | & \underbrace{\beta_2 w_2(P_{m,n})} & : & \delta_2 \\ & & & & x_i^n(P_{m,n}) & & \end{pmatrix}, \quad (14)$$

where $b'_i(P_{m,n}) = b_i(P_{m,n}) + w_i(P_{m,n})$ and $b'_i(P_n \setminus P_{m,n}) = b_i(P_n \setminus P_{m,n})$ for $i = 1, 2$, and

$$y_i^n(P_{m,n}) = \begin{pmatrix} \beta_1 & w_1(P_{m,n}) \\ \beta_2 & w_2(P_{m,n}) \end{pmatrix} \quad (15)$$

is the updated message label for \mathcal{D}_n .

The fault labels y_f^n and x_f^n are the same since by assumption the fault types for each module are disjoint and the transitions removing tokens from or putting tokens into the common places are labeled with observable events. \square

VI. MONOLITHIC PETRI NET DIAGNOSERS

A brief review of the section on monolithic Petri net diagnosers in [15] is required for completeness of the results presented in Section VII that follows. If the set of place-bordered nets is a singleton, then we say that the system to be diagnosed is monolithic and the corresponding diagnoser is a monolithic Petri net diagnoser. Monolithic Petri net diagnosers have states that do not carry message labels since those are not needed in that case. We may form a monolithic system by combining the modules in a set of place-bordered nets. Formally, we have

$$\mathcal{C}_{\mathcal{S}} = (\langle P, T, A, w \rangle, \Sigma, l, x_0),$$

where $\mathcal{S} = \{(\mathcal{M}_m, \mathcal{P}_m) : m = 1, 2, \dots, M\}$. We form the set of places of the monolithic system as $P = \bigcup_{m \in \{1, 2, \dots, M\}} P_m$. Similarly for T, A, Σ . For each module $\mathcal{M}_m \in \mathcal{S}$, we have $w|_{A_m} = w_m, l|_{T_m} = l_m$, and $x_0(P_m) = x_0^m$. We use the monolithic Petri diagnoser of $\mathcal{C}_{\mathcal{S}}$ by $\mathcal{C}_{d,\mathcal{S}}$.

VII. CORRECTNESS RESULTS

In this section, we present correctness results for DDC- M . The proofs of the results in this section are given in the appendix. The following lemma shows that, if for some rows of the diagnoser states of two place-bordered modules the message labels are the same, then for those rows the state information of the common places between those two modules must be the same. Later in the section, we use the result of Lemma 1 to define the *merge* operation that leads to the main result of the section.

Lemma 1: Given the set of place-bordered nets \mathcal{S} , and the set of corresponding diagnosers $\mathcal{S}_{\mathcal{D}}$, let $\{x_{d,r}^m : m = 1, 2, \dots, M\}$ be the set of diagnoser states of the modules $\mathcal{D}_m \in \mathcal{S}_{\mathcal{D}}$ after the sequence $\sigma_{o1}\sigma_{o2}\dots\sigma_{oR}$ of observable events where $R \in \mathbb{N}$. For all $\mathcal{D}_n \in \mathcal{S}_{\mathcal{D}}$ such that $P_{m,n} \neq \emptyset$ if $x_{l,R}^m(i, P_{m,n}) = x_{l,R}^n(j, P_{m,n})$ for some rows i_m and i_n , then $x_{s,R}^m(i_m, P_{m,n}) = x_{s,R}^n(i_n, P_{m,n})$.

In view of Lemma 1, we define an operation called *merge* that combines the diagnoser states of the modules.

Definition 1 (Merge): Given the set of place-bordered nets \mathcal{S} and the set of corresponding diagnosers $\mathcal{S}_{\mathcal{D}}$, let x_d^m be the diagnoser state of $\mathcal{D}_m \in \mathcal{S}_{\mathcal{D}}$ for $m = 1, 2, \dots, M$ after some sequence of observable events. We define the merge operation on these states recursively as follows:

- 1) Merge of two diagnoser states, $\mathcal{D}_m, \mathcal{D}_n \in \mathcal{S}_{\mathcal{D}}$. There are two cases:

- a) $P_{m,n} = \emptyset$. In this case for all rows i_m, i_n of x_d^m and x_d^n , respectively,

$$(x_s^m(i_m, P_m), x_s^n(i_n, P_n) \mid x_f^m x_f^n) \\ \in \text{Merge}(x_d^m, x_d^n)(P_m \cup P_n \mid \Delta_{f,m} \cup \Delta_{f,n}).$$

- b) $P_{m,n} \neq \emptyset$. In this case for all rows i_m, i_n of x_d^m and x_d^n , respectively, such that $x_l^m(i_m, P_{m,n}) = x_l^n(i_n, P_{m,n})$,

$$(x_s^m(i_m, P_m), x_s^n(i_n, P_n \setminus P_m) \mid x_f^m x_f^n) \\ \in \text{Merge}(x_d^m, x_d^n)(P_m \cup P_n \mid \Delta_{f,m} \cup \Delta_{f,n}).$$

- 2) Let $\mathcal{D}_m, \mathcal{D}_n, \mathcal{D}_q \in \mathcal{S}_{\mathcal{D}}$. Then,

$$\text{Merge}(x_d^m, x_d^n, x_d^q) = \text{Merge}(\text{Merge}(x_d^m, x_d^n), x_d^q).$$

The intuition behind the merge of diagnoser states of place-bordered modules is to form composed states by concatenating rows whose message labels match (case (1)(b)). This constraint is waved when the modules are not coupled, since all combinations of rows are possible (case (1)(a)).

In the rest of this section, we present the relations between the monolithic system formed by combining the modules in a set of place-bordered nets and the

distributed diagnosis system where a diagnoser is attached to each place-bordered net and communication is allowed between the diagnosers.

In the following lemma, we state that if a sequence of observable events is feasible in the monolithic system, then the merge of the diagnoser states of the place-bordered modules will not result in an empty set.

Lemma 2: Given the set of place-bordered nets \mathcal{S} , and the set of corresponding diagnosers $\mathcal{S}_{\mathcal{D}}$, let $\{x_{d,r}^m : m = 1, 2, \dots, M\}$ be the set of diagnoser states of the modules $\mathcal{D}_m \in \mathcal{S}_{\mathcal{D}}$ and $\mathcal{C}_{\mathcal{S}}$ be the the monolithic Petri net formed by combining the modules in \mathcal{S} where $r \in \mathbb{N}$. If the sequence of observable events $\sigma_{o1}\sigma_{o2}\dots\sigma_{or}$ is feasible in $\mathcal{C}_{\mathcal{S}}$, then $\text{Merge}(x_{d,r}^m : \mathcal{D}_m \in \mathcal{S}_{\mathcal{D}}) \neq \emptyset$.

The following theorem states that DDC- M is correct in the sense that the merge operation recovers the corresponding monolithic diagnoser state. That is, when the token distribution of a set of common places changes, the change in the token distribution and the past history along which the change has occurred is sent via message labels. Thus, in a way, message labels not only record the history of changes but also create a common knowledge of shared history among the modules in the system. Then, if we concatenate rows whose message labels match as it is defined by the merge operation, we combine exactly the rows with the very same history and form the monolithic diagnoser state.

Theorem 1: Given the set of place-bordered nets \mathcal{S} , and the set of corresponding diagnosers $\mathcal{S}_{\mathcal{D}}$, let $\{x_{d,r}^m : m = 1, 2, \dots, M\}$ be the set of diagnoser states of the modules $\mathcal{D}_m \in \mathcal{S}_{\mathcal{D}}$ and $\mathcal{X}_{d,r}$ be the set of states of the monolithic diagnoser state $x_{d,r}$ of $\mathcal{C}_{\mathcal{S}}$ after observation of the feasible sequence $\sigma_{o1}\sigma_{o2}\dots\sigma_{or}$ where $r \in \mathbb{N}$. Then,

$$\text{Merge}(x_{d,r}^m : \mathcal{D}_m \in \mathcal{S}_{\mathcal{D}}) = \mathcal{X}_{d,r}.$$

VIII. IMPLEMENTATION OF DDC- M : FIXED-SIZE MESSAGE LABELS

The version of Algorithm DDC- M presented in Section V recovers the monolithic diagnosis information at the cost of communication and growing message labels. The size of the message label is bounded by the number of common places and the number of observable events executed by the system. Thus, observations of longer sequences of events result in longer message labels. There are several ways to reduce the communication overhead by reducing the size of the message labels while still recovering the monolithic diagnosis information. In this regard, we now present an encoding-based method which serves this purpose and results in fixed-size message labels. We first describe the structure of the message labels and how the encoding makes it possible to have

fixed-size messages and message labels. Secondly, we update the DDC- M algorithm to reflect the changes in the messages and message labels. We continue with an example showing the implementation of the updated DDC- M algorithm. We conclude the section by proving the correctness of the updated algorithm in the sense that the merge operation still recovers the monolithic diagnoser state after observation of a sequence of events.

Suppose that the set of place-bordered nets \mathcal{S} is the system to be diagnosed and $\sigma_{o1}\sigma_{o2}\dots\sigma_{oR}$ is the sequence of events observed. Let $\mathcal{M}_m, \mathcal{M}_n \in \mathcal{S}$ be two place-bordered nets with corresponding common places $P_{m,n}$ where $P_{m,n} \neq \emptyset$. We define the set $\Omega_{m,n}^R$ of words such that each word $\omega \in \Omega_{m,n}^R$ is a combination of elements from the finite set $C_{m,n} = \{W_m(t, P_{m,n}) : t \in T_m\}$ and the length of the word is at most R . Formally, we have

$$\Omega_{m,n}^R = \{\omega_1\omega_2\dots\omega_k : \forall 1 \leq i \leq k, \omega_i \in C_{m,n} \text{ and } 1 \leq k \leq R \text{ where } R \in \mathbb{N}\}. \quad (16)$$

The elements of $C_{m,n}$ are vectors of size $|P_{m,n}|$ and correspond to all possible changes in the token distribution of the common places upon firing of a transition. The set $C_{m,n}$ is finite since the arcs removing tokens from or putting tokens into the common places are of finite weight, and there is a finite number of observable transitions removing tokens from or putting tokens into the common places. Thus, each word $\omega \in \Omega_{m,n}^R$ is a possible combination of changes that may occur in the common places upon observation of a sequence of R events. If $x_{l,R}^m$ is the message label after observation of a sequence of R events, then each row of $x_{l,R}^m$ corresponds to a word in the set $\Omega_{m,n}^R$.

Our goal is to find a function $g^R : \Omega_{m,n}^R \rightarrow \mathbb{N}$ for all $R \in \mathbb{Z}_{>0}$ such that g^R is injective. One such function is the enumeration of the different words in $\Omega_{m,n}^R$, starting with 1, that corresponds to the enumeration of the different rows of $x_{l,R}^m$. We describe such an injective enumeration in Definition 2. Since our goal is to enumerate the different rows of a message label and message labels are matrices, we define the enumeration of different rows of a matrix instead of different elements of a set. When we write $En(x_{l,R}^m)$, we mean the enumeration of the different rows of $x_{l,R}^m$ as in Definition 2.

Definition 2 (Enumeration): Given a matrix A , we denote by $A(i)$ the i^{th} row of A . Then, we define En as follows:

$$1) \text{ } En(A(1)) = 1; \\ 2) \text{ For all } i \in \{2, 3, \dots, \# \text{ of rows of } A\}, \\ En(A(i)) = \begin{cases} En(A(j)), \exists j \in \{1, 2, \dots, i-1\} \\ \quad \text{such that } A(j) = A(i), \\ 1 + \max\{En(A(j)) : 1 \leq j < i\}, \\ \text{otherwise.} \end{cases}$$

We update Algorithm 1 to 3 and Algorithm 2 to 4 to account for fixed-size message labels. The updated algorithms evolve the message labels consistent with the enumeration function described in Definition 2.

The formal statement of Algorithms 3 and 4 is given below. In Algorithm 4, $Mesg_{m,n}.Sfx(i, 1)$ denotes the columns of $Mesg_{m,n}.Sfx$ that correspond to the changes in the token distribution of the common places, and $Mesg_{m,n}.Sfx(i, 2)$ denotes the column that corresponds to the (new) enumeration.

Algorithm 3 Distributed Diagnosis with Communication with Fixed-Size Message Labels

```

1: Upon occurrence of an observable event  $\sigma_{or}$ 
2: Find  $\mathcal{M}_m$  such that  $\sigma_{or} \in \Sigma_m$ ,
3:  $x_{d,r}^m \leftarrow J_{d,m}(x_{d,r-1}^m, \sigma_{or})$ ,
4:  $x_{d,r}^m \leftarrow z_{d,r}^m$ ,
5: for all  $\mathcal{Q}_n \in \mathcal{S}_{\mathcal{Q}}$  such that  $P_{m,n} \neq \emptyset$  do
6:    $x_{l,r}^m(P_{m,n}) \leftarrow En(z_{l,r}^m(P_{m,n}))$ ,
7:   if  $\{W(P_{m,n}, t) \mid t \in B_m(x_{d,r-1}^m, \sigma_{or})\} \neq \{\emptyset\}$  then
8:      $Mesg_{m,n} \leftarrow \{ \}$ ,
9:     for all  $j=1:\#$  of rows of  $x_{l,r}^m(P_{m,n})$  do
10:       $Mesg_{m,n}.Pfx(j) \leftarrow z_{l,r}^m(j, P_{m,n}).Pfx$ ,
11:       $Mesg_{m,n}.Sfx(j) \leftarrow (z_{l,r}^m(j, P_{m,n}).Sfx \ x_{l,r}^m(j, P_{m,n}))$ ,
12:       $Mesg_{m,n}(j) \leftarrow (Mesg_{m,n}.Pfx(j) \ Mesg_{m,n}.Sfx(j))$ ,
13:    end for
14:    Send all different rows of  $Mesg_{m,n}$ ,
15:     $x_{d,r}^m \leftarrow UDSC(x_{d,r-1}^m, Mesg_{m,n})$ ,
16:  end if
17: end for
```

Algorithm 4 Update of Diagnoser State upon Communication with Fixed-Size Message Labels

```

Require:  $x_{d,r-1}^m, Mesg_{m,n}$ 
1:  $X_{d,r}^m \leftarrow \{ \}$ ,
2: for all  $i = 1$  : Number of rows of  $Mesg_{m,n}.Pfx$  do
3:   for all  $j = 1$  : Number of rows of  $x_{l,r-1}^m(P_{m,n})$  do
4:     if  $Mesg_{m,n}.Pfx(i) == x_{l,r-1}^m(j, P_{m,n})$  then
5:        $y_s(P_{m,n}) \leftarrow x_{s,r-1}^m(j, P_{m,n}) + Mesg_{m,n}.Sfx(i, 1)$ ,
6:        $y_s(P(n) \setminus P_{m,n}) \leftarrow x_{s,r-1}^m(j, P_n \setminus P_{m,n})$ ,
7:        $y_f \leftarrow x_f^m(j)$ ,
8:        $y_l(P_{m,n}) \leftarrow Mesg_{m,n}.Sfx(i, 2)$ ,
9:       for all  $\mathcal{Q}_q \in (\mathcal{S}_{\mathcal{Q}} \setminus \mathcal{Q}_m)$  such that  $P_{n,q} \neq \emptyset$  do
10:         $y_l(P_{n,q}) \leftarrow x_{l,r-1}^m(j, P_{n,q})$ 
11:      end for
12:       $X_{d,r}^m \leftarrow X_{d,r}^m \cup [y_s | y_f | y_l]$ 
13:    end if
14:  end for
15:  $UDSC(x_{d,r-1}^m, Mesg_{m,n}) \leftarrow$  Listing of the set  $X_{d,r}^m$ 
```

Theorem 2: Theorem 1 is valid for the diagnoser states obtained under Algorithms 3 and 4.

The proof of Theorem 2 is given in the appendix. The key idea that results in the fixed-size message labels is that the next state in a Petri net is uniquely found by the current state and the changes in the token distribution of the places. We now consider how this idea is implemented while message labels are created. In Algorithm 1, we form the message label of the next diagnoser state by appending the changes on the common

places to the message labels of the current diagnoser state. However, in Algorithm 3, we uniquely encode the message label found by the diagnoser state transition function and the encoded message label is the message label of the next diagnoser state. That is, the message label of the next diagnoser state is a bijective function of the message label of the current diagnoser state and the changes on the common places. Algorithms 2 and 4 do not differ in structure as do Algorithms 1 and 3. Algorithm 4 correctly updates the diagnoser states of the neighboring states because we use a bijective function to encode the message label.

In the following example, we illustrate the notion and notations presented in this section while comparing the steps of Algorithms 3 and 4 to 1 and 2.

Example 3: In Example 2, we derived the diagnoser states when we run Algorithms 1 and 2. In this example, we consider the same setting as in Example 2, however, we derive the diagnoser states when we run Algorithms 3 and 4 instead. The state and fault labels of the diagnoser states in this case are the same as the state and fault labels given in Example 2. However, the message labels and messages sent are changed. In the following, we go over the steps of Algorithms 3 and 4 to find the changes in the message labels.

Suppose that \mathcal{M}_m and \mathcal{M}_n are two coupled modules in \mathcal{S} . The diagnoser states x_d^m and x_d^n of \mathcal{D}_m and \mathcal{D}_n , respectively, obtained under Algorithms 3 and 4 have same abbreviations as x_d^m in Equation (10) and x_d^n in Equation (11), respectively.

In this example, we focus on the message labels between \mathcal{D}_m and \mathcal{D}_n . We put the sign $*$ for the message labels for all modules $\mathcal{M}_{n'} \in \mathcal{S}$ coupled with \mathcal{M}_m except \mathcal{M}_n and for all modules $\mathcal{M}_{m'} \in \mathcal{S}$ coupled with \mathcal{M}_n except \mathcal{M}_m .

Suppose that the event $\sigma_o \in \Sigma_{o,m}$ is observed, then the intermediate diagnoser state $z_d^m = f_{d,m}(x_d^m, \sigma_o)$ is found as follows

$$z_d^m = \left(\begin{array}{c|c|c} \cdot & \cdot & \varphi_1 \ w_1(P_{m,n}) \ : \ \cdot \\ \cdot & \cdot & \varphi_2 \ w_2(P_{m,n}) \ : \ \cdot \\ \cdot & \cdot & \varphi_1 \ w_1(P_{m,n}) \ : \ \cdot \\ \cdot & \cdot & \varphi_2 \ w_2(P_{m,n}) \ : \ \cdot \end{array} \right) \quad (17)$$

$\underbrace{\hspace{1.5cm}}_{x_s^m} \quad \underbrace{\hspace{1.5cm}}_{x_f^m} \quad \underbrace{\hspace{1.5cm}}_{z_l^m(P_{m,n})}$

Suppose that the encoding of the message label is as follows

$$En(z_l^m(P_{m,n})) = \begin{pmatrix} 1 \\ 2 \\ 1 \\ 2 \end{pmatrix}. \quad (18)$$

Then, the diagnoser state y_d^m of \mathcal{D}_m upon observation σ_o is constructed as (the reader is encouraged to compare

to the diagnoser state in Equation (12) obtained under Algorithm 3)

$$y_d^m = \left(\begin{array}{c|c|c} a_1+w_1 & h_1 & 1 \ : \ \cdot \\ a_2+w_2 & h_2 & 2 \ : \ \cdot \\ a_1+w_1+\overline{w_1} & h'_1 & 1 \ : \ \cdot \\ a_2+w_2+\overline{w_2} & h'_2 & 2 \ : \ \cdot \end{array} \right). \quad (19)$$

The message sent from \mathcal{D}_m to \mathcal{D}_n is

$$Msg_{m,n} = \left(\begin{array}{c|c|c} \varphi_1 & w_1(P_{m,n}) & 1 \\ \varphi_2 & w_2(P_{m,n}) & 2 \end{array} \right),$$

$Msg_{m,n}.Pfx \quad Msg_{m,n}.Sfx(1) \quad Msg_{m,n}.Sfx(2)$

Upon reception of the message \mathcal{D}_n updates x_d^n to y_d^n based on the message from \mathcal{D}_m (as defined in Algorithm 4) as follows (the reader is encouraged to compare to the diagnoser state in Equation (14) obtained under Algorithm 4)

$$y_d^n = \left(\begin{array}{c|c|c} b'_1 & k_1 & 1 \ : \ \cdot \\ b'_2 & k_2 & 2 \ : \ \cdot \\ \cdot & \cdot & \cdot \ : \ \cdot \end{array} \right). \quad (20)$$

$y_s^n \quad y_f^n \quad y_l^n(P_{m,n})$

□

IX. CASE STUDY

We developed a software implementation of DDC- M and of the merge operation. The software interacts with *GraphViz* developed by AT&T to visualize the labeled Petri nets, diagnoser states (including the state, fault and message information) and dynamics of the Petri nets and the algorithms (if communications occur among modules, which module communicates with which module, list of events enabled from the diagnoser states, etc.). All the analysis results of the examples in this section are performed using the software tool.

In the following, we study an example of an *Heating, Ventilation and Air-Conditioning System*. We consider the valve, pump and load models shown in Figs. 4, 5 and 6, respectively. Together they form the set of place-bordered labeled Petri nets that constitute the overall system. The sets of events of these place-bordered nets are disjoint, hence, so are the sets of transitions. The place-bordered nets of the valve, pump and load are coupled with each other through common places. For example, place c_1 appears in both the valve and load model in Figs. 4 and 6, respectively. Figure 7 shows the coupling between the individual place-bordered nets for the overall system. For all the labeled Petri nets in this paper, the filled transitions are labeled with unobservable events.

The set of events and the abbreviations in the Fig. 4 to 6 for the events are as follows: $\Sigma_{o,1} =$

$\{close_valve(cv), open_valve(ov), stuck_open_1(so1), stuck_open_2(so2), stuck_closed_1(sc1), stuck_closed_2(sc2)\}$, $\Sigma_{o,2} = \{start_pump(st), stop_pump(sp), pump_failed_on_1(fn1), pump_failed_on_2(fn2), pump_failed_off_1(fo1), pump_failed_off_2(fo2)\}$, $\Sigma_{o,3} = \{set_point_decrease(spd), set_point_increase(spi), failed_off(foff)\}$.

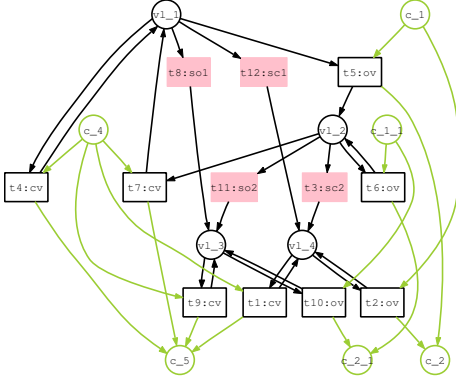


Fig. 4. Place-bordered net: Module#1 (valve).

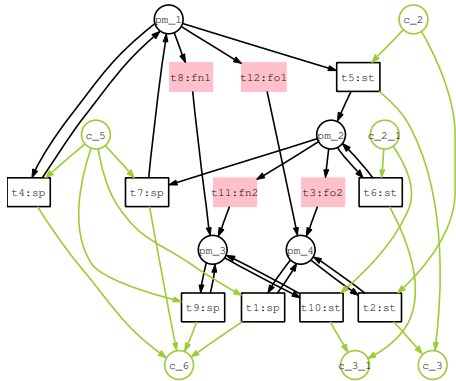


Fig. 5. Place-bordered net: Module#2 (pump).

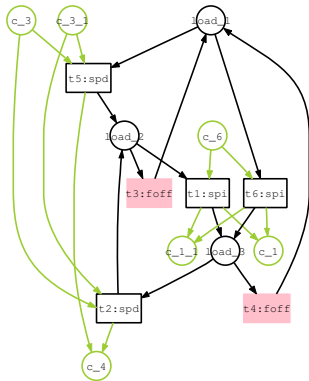


Fig. 6. Place-bordered net: Module#3 (load).

Suppose that initially there is only one token at each of the following places: c_1 , c_{1_1} , vl_1 , pm_1 and $load_1$.

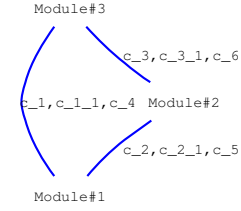


Fig. 7. Common places between the modules.

Then, the initial diagnoser states of the modules are as follows as defined by the diagnoser state transition function in Equations (5) to (7).

The initial diagnoser state of \mathcal{D}_1 (the diagnoser for *Module#1*) is shown in Equation 21, where each digit in the rows of $x_{s,0}^1$ correspond to the number of tokens in a place of \mathcal{D}_1 , and each digit in the rows of $x_{f,0}^1$ corresponds a fault type of \mathcal{D}_1 . The ordering of the digits in $x_{s,0}^1$ is as follows: $c_1, c_{1_1}, c_2, c_{2_1}, c_4, c_5, vl_1, vl_2, vl_3, vl_4$. The ordering of digits in $x_{f,0}^1$ is $F1$ and $F2$, respectively, where the event sets for the fault types are as follows: $\Sigma_{F1,1} = \{stuck_open_1(so1), stuck_open_2(so2)\}$, $\Sigma_{F2,1} = \{stuck_closed_1(sc1), stuck_closed_2(sc2)\}$.

$$x_{d,0}^1 = \left(\begin{array}{c|c} 1100001000 & 00 \\ 1100000010 & 10 \\ 1100000001 & 01 \end{array} \right), \quad x_{d,0}^2 = \left(\begin{array}{c|c} 0000001000 & 00 \\ 0000000010 & 10 \\ 0000000001 & 01 \end{array} \right),$$

$$x_{d,0}^3 = \left(\begin{array}{c|c} 110000100 & 0 \end{array} \right), \quad (21)$$

The initial diagnoser state of \mathcal{D}_2 (the diagnoser for *Module#2*) is shown in Equation 21, where each digit in the rows of $x_{s,0}^2$ corresponds to the number of tokens in a place of \mathcal{D}_2 , and each digit in the rows of $x_{f,0}^2$ corresponds a fault type of \mathcal{D}_2 . The ordering of the digits in $x_{s,0}^2$ is as follows: $c_2, c_{2_1}, c_3, c_{3_1}, c_5, c_6, pm_1, pm_2, pm_3, pm_4$. The ordering of digits in $x_{f,0}^2$ is $F1$ and $F2$, respectively, where the event sets for the fault types are as follows: $\Sigma_{F1,2} = \{pump_failed_on_1(fn1), pump_failed_on_2(fn2)\}$, $\Sigma_{F2,2} = \{pump_failed_off_1(fo1), pump_failed_off_2(fo2)\}$,

The initial diagnoser state of \mathcal{D}_3 (the diagnoser for *Module#3*) is shown in Equation 21, where each digit in the rows of $x_{s,0}^3$ corresponds to the number of tokens in a place of \mathcal{D}_3 , and each digit in the rows of $x_{f,0}^3$ corresponds a fault type of \mathcal{D}_3 . The ordering of the digits in $x_{s,0}^3$ is as follows: $c_1, c_{1_1}, c_3, c_{3_1}, c_4, c_6, load_1, load_2, load_3$. The ordering of digits in $x_{f,0}^3$ is $F1$ where the event set for the

fault type is as follows: $\Sigma_{F1,3} = \{failed_off(foff)\}$.

The initial diagnoser states do not have message labels by assumption. Thus, the diagnoser states in (21), (??) and (??) have state and fault type information only.

The only observable event enabled is *open_valve*. If the event *open_valve* is observed, then applying Algorithm 1, *Module#1* finds the next diagnoser state using the diagnoser state transition function and sends messages to *Module#2* and *Module#3*. Upon reception of the messages, *Module#2* and *Module#3* update their current diagnoser states according to Algorithm 2. Overall, the diagnoser states obtained by Algorithms 1 and 2 are presented in the following. The diagnoser state for \mathcal{D}_1 is

$$x_{d,1}^1 = \begin{pmatrix} 0110000001 & | & 01 & | & 100 & : & -100 \\ 0110000010 & | & 10 & | & 100 & : & -100 \\ 0110000100 & | & 00 & | & 100 & : & -100 \\ \underbrace{1001000010}_{x_{s,1}^1} & | & \underbrace{10}_{x_{f,1}^1} & | & \underbrace{010}_{x_{l,1}^1(P_{1,2})} & : & \underbrace{0-10}_{x_{l,1}^1(P_{1,3})} \end{pmatrix}, \quad (22)$$

where each digit (with the minus sign) in the rows of the message labels $x_{l,1}^1(P_{1,2})$ and $x_{l,1}^1(P_{1,3})$ corresponds to the difference between the number of tokens put into and removed from a common place. The ordering of digits for the message labels are as follows: c_2, c_{2-1}, c_5 for $x_{l,1}^1(P_{1,2})$, and c_1, c_{1-1}, c_4 for $x_{l,1}^1(P_{1,3})$.

Upon reception of the message from \mathcal{D}_1 after the observation of *open_valve*, the diagnoser state for \mathcal{D}_2 is updated to (by following the steps of Algorithm 2)

$$x_{d,1}^2 = \begin{pmatrix} 0100000001 & | & 01 & | & 010 & : & \\ 0100000010 & | & 10 & | & 010 & : & \\ 0100001000 & | & 00 & | & 010 & : & \\ 1000000001 & | & 01 & | & 100 & : & \\ 1000000010 & | & 10 & | & 100 & : & \\ \underbrace{1000001000}_{x_{s,1}^2} & | & \underbrace{00}_{x_{f,1}^2} & | & \underbrace{100}_{x_{l,1}^2(P_{2,1})} & : & \underbrace{}_{x_{l,1}^2(P_{2,3})} \end{pmatrix}, \quad (23)$$

where each digit (with the minus sign) in the rows of the message labels $x_{l,1}^2(P_{2,1})$ and $x_{l,1}^2(P_{2,3})$ corresponds to the difference between the number of tokens put into and removed from a common place. The ordering of digits for the message labels are as follows: c_2, c_{2-1}, c_5 for $x_{l,1}^2(P_{2,1})$, and c_3, c_{3-1}, c_6 for $x_{l,1}^2(P_{2,3})$.

Upon reception of the message from \mathcal{D}_1 after the observation of *open_valve*, the diagnoser state for \mathcal{D}_3 is

$$x_{d,1}^3 = \begin{pmatrix} 010000100 & | & 0 & | & -100 & : & \\ \underbrace{100000100}_{x_{s,1}^3} & | & \underbrace{0}_{x_{f,1}^3} & | & \underbrace{0-10}_{x_{l,1}^3(P_{3,1})} & : & \underbrace{}_{x_{l,1}^3(P_{3,2})} \end{pmatrix}, \quad (24)$$

where each digit (with the minus sign) in the rows of the message labels $x_{l,1}^3(P_{2,1})$ and $x_{l,1}^3(P_{3,1})$ corresponds to the

difference between the number of tokens put into and removed from a common place. The ordering of digits for the message labels are as follows: c_1, c_{1-1}, c_4 for $x_{l,1}^3(P_{3,1})$, and c_3, c_{3-1}, c_6 for $x_{l,1}^3(P_{3,2})$.

The next enabled observable event is *start_pump*. Upon its occurrence, *Module#2* finds the next diagnoser state using the diagnoser state transition function and sends messages to *Module#1* and *Module#3*. After the observation of *start_pump* and the diagnoser state updates triggered by the reception of messages, the state with fault information and message labels of the new diagnoser states are as follows:

$$x_{d,2}^1 = \begin{pmatrix} 0100000001 & | & 01 & | & 100-100 & : & -100 \\ 0100000010 & | & 10 & | & 100-100 & : & -100 \\ 0100000100 & | & 00 & | & 100-100 & : & -100 \\ 1000000010 & | & 10 & | & 0100-10 & : & 0-10 \end{pmatrix} \quad (25)$$

$$x_{d,2}^2 = \begin{pmatrix} 0001000010 & | & 10 & | & 0100-10 & : & 010 \\ 0010000001 & | & 01 & | & 100-100 & : & 100 \\ 0010000010 & | & 10 & | & 100-100 & : & 100 \\ 0010000100 & | & 00 & | & 100-100 & : & 100 \end{pmatrix} \quad (26)$$

$$x_{d,2}^3 = \begin{pmatrix} 010100100 & | & 0 & | & -100 & : & 010 \\ 011000100 & | & 0 & | & -100 & : & 100 \\ 100100100 & | & 0 & | & 0-10 & : & 010 \\ 101000100 & | & 0 & | & 0-10 & : & 100 \end{pmatrix} \quad (27)$$

Upon the occurrence of the next observable event the algorithm will proceed in the same manner to update the respective diagnoser states.

An examination of the fault labels in the corresponding columns of the above diagnoser states reveals that: (i) $x_{d,0}^1, x_{d,1}^1$ and $x_{d,2}^1$ are both $F_{1,1}$ – *uncertain* (*stuck_open_1* or *stuck_open_2* could have happened but we do not know for sure) and $F_{2,1}$ – *uncertain*, (ii) $x_{d,0}^2, x_{d,1}^2$ and $x_{d,2}^2$ are both $F_{1,2}$ – *uncertain* and $F_{2,2}$ – *uncertain*, and (iii) $x_{d,0}^3, x_{d,1}^3$ and $x_{d,2}^3$ are normal.

We now consider the case of fixed-size message labels. Suppose that we observe the very same sequence of events which starts with the event *open_valve* followed by *start_pump*, and we now run Algorithm 3 instead of 1 and Algorithm 4 instead of 2. The state and fault labels of the diagnoser states in this case are the same with the state and fault labels given in Equations (21) to (27). However, the message labels and messages sent are changed. In the following, we go over the steps of Algorithms 3 and 4 to find the changes in the message labels.

The message labels of the initial diagnoser states are all equal to 1 by construction. Upon observation of the

event *open_valve* (executed by \mathcal{M}_1), the intermediate diagnoser state $z_{d,1}^1 = f_{d,1}(x_{d,0}^1, \text{open_valve})$ is

$$z_{d,1}^1 = \left(\begin{array}{c|c|c|c} \cdot & \cdot & 1\ 100 & 1\ -100 \\ \cdot & \cdot & 1\ 100 & 1\ -100 \\ \cdot & \dots & 1\ 100 & 1\ -100 \\ \hline \underbrace{\cdot} & \underbrace{\cdot} & \underbrace{1\ 010} & \underbrace{1\ 0-10} \\ x_{s,1}^1 & x_{f,1}^1 & z_{l,1}^1(P_{1,2}) & z_{l,1}^1(P_{1,3}) \end{array} \right), \quad (28)$$

The message labels for the diagnoser state $x_{d,1}^1$ are $x_{l,1}^1(P_{1,2}) = En(z_{l,1}^1(P_{1,2}))$ and $x_{l,1}^1(P_{1,3}) = En(z_{l,1}^1(P_{1,3}))$ for \mathcal{D}_2 and \mathcal{D}_3 , respectively. Thus, the diagnoser state in the case of fixed-size message labels (compare to one in 28) is found as

$$x_{d,1}^1 = \left(\begin{array}{c|c|c|c} \cdot & \cdot & 1 & 1 \\ \cdot & \cdot & 1 & 1 \\ \cdot & \cdot & 1 & 1 \\ \hline \underbrace{\cdot} & \underbrace{\cdot} & \underbrace{2} & \underbrace{2} \\ x_{s,1}^1 & x_{f,1}^1 & x_{l,1}^1(P_{1,2}) & x_{l,1}^1(P_{1,3}) \end{array} \right). \quad (29)$$

The messages sent by \mathcal{D}_1 are as follows:

$$Mesg_{1,2} = \left(\begin{array}{c|c|c} \underbrace{1} & \underbrace{100} & \underbrace{1} \\ \underbrace{1} & \underbrace{010} & \underbrace{2} \end{array} \right),$$

$$Mesg_{1,2}.Pfx \quad Mesg_{1,2}.Sfx(1) \quad Mesg_{1,2}.Sfx(2)$$

$$Mesg_{1,3} = \left(\begin{array}{c|c|c} \underbrace{1} & \underbrace{-100} & \underbrace{1} \\ \underbrace{1} & \underbrace{0-10} & \underbrace{2} \end{array} \right),$$

$$Mesg_{1,3}.Pfx \quad Mesg_{1,3}.Sfx(1) \quad Mesg_{1,3}.Sfx(2)$$

Upon reception of the message the diagnoser states of the neighbor modules are updated as defined by Algorithm 4. Then, the diagnoser states of \mathcal{D}_2 and \mathcal{D}_3 are as follows:

$$x_{d,2}^2 = \left(\begin{array}{c|c|c|c} \cdot & \cdot & 2 & 1 \\ \cdot & \cdot & 2 & 1 \\ \cdot & \cdot & 2 & 1 \\ \cdot & \cdot & 1 & 1 \\ \cdot & \cdot & 1 & 1 \\ \hline \underbrace{\cdot} & \underbrace{\cdot} & \underbrace{1} & \underbrace{1} \\ x_{s,2}^2 & x_{f,2}^2 & x_{l,2}^2(P_{2,1}) & x_{l,2}^2(P_{2,3}) \end{array} \right), \quad (30)$$

$$x_{d,3}^3 = \left(\begin{array}{c|c|c|c} \cdot & \cdot & 1 & 1 \\ \cdot & \cdot & 2 & 1 \\ \hline \underbrace{\cdot} & \underbrace{\cdot} & \underbrace{2} & \underbrace{1} \\ x_{s,1}^1 & x_{f,1}^1 & x_{l,1}^1(P_{2,1}) & x_{l,1}^1(P_{3,2}) \end{array} \right). \quad (31)$$

Upon observation of the event *start_pump* executed by \mathcal{D}_2 , the intermediate diagnoser state, $z_{d,2}^2 = f_{d,2}(z_{d,2}^2, \text{start_pump})$, is found as:

$$z_{d,2}^2 = \left(\begin{array}{c|c|c|c} \cdot & \cdot & 2\ 0-10 & 1\ 010 \\ \cdot & \cdot & 1\ -100 & 1\ 100 \\ \cdot & \cdot & 1\ -100 & 1\ 100 \\ \hline \underbrace{\cdot} & \underbrace{\cdot} & \underbrace{1\ -100} & \underbrace{1\ 100} \\ x_{s,1}^2 & x_{f,1}^2 & z_{l,1}^2(P_{2,1}) & z_{l,1}^2(P_{2,3}) \end{array} \right), \quad (32)$$

The message labels for the diagnoser state $x_{d,1}^2$ are $x_{l,1}^2(P_{1,2}) = En(z_{l,1}^2(P_{1,2}))$ and $x_{l,1}^2(P_{1,3}) = En(z_{l,1}^2(P_{1,3}))$ for \mathcal{D}_2 and \mathcal{D}_3 , respectively. Thus, the diagnoser state in the case of fixed-size message labels is found as

$$x_{d,2}^2 = \left(\begin{array}{c|c|c|c} \cdot & \cdot & 1 & 1 \\ \cdot & \cdot & 2 & 2 \\ \cdot & \cdot & 2 & 2 \\ \hline \underbrace{\cdot} & \underbrace{\cdot} & \underbrace{2} & \underbrace{2} \\ x_{s,1}^2 & x_{f,1}^2 & x_{l,1}^2(P_{2,1}) & x_{l,1}^2(P_{2,3}) \end{array} \right), \quad (33)$$

The messages sent by \mathcal{D}_2 are as follows:

$$Mesg_{2,1} = \left(\begin{array}{c|c|c} \underbrace{2} & \underbrace{0-10} & \underbrace{1} \\ \underbrace{1} & \underbrace{-100} & \underbrace{2} \end{array} \right),$$

$$Mesg_{2,1}.Pfx \quad Mesg_{2,1}.Sfx(1) \quad Mesg_{2,1}.Sfx(2)$$

$$Mesg_{2,3} = \left(\begin{array}{c|c|c} \underbrace{1} & \underbrace{010} & \underbrace{1} \\ \underbrace{1} & \underbrace{100} & \underbrace{2} \end{array} \right),$$

$$Mesg_{2,3}.Pfx \quad Mesg_{2,3}.Sfx(1) \quad Mesg_{2,3}.Sfx(2)$$

Upon reception of the message the diagnoser states of the neighbor modules are updated as defined by Algorithm 4. Then, the diagnoser states of \mathcal{D}_1 and \mathcal{D}_3 are as follows:

$$x_{d,2}^1 = \left(\begin{array}{c|c|c|c} \cdot & \cdot & 2 & 1 \\ \cdot & \cdot & 2 & 1 \\ \cdot & \cdot & 2 & 1 \\ \hline \underbrace{\cdot} & \underbrace{\cdot} & \underbrace{1} & \underbrace{2} \\ x_{s,2}^1 & x_{f,2}^1 & x_{l,2}^1(P_{1,2}) & x_{l,2}^1(P_{1,3}) \end{array} \right). \quad (34)$$

$$x_{d,3}^3 = \left(\begin{array}{c|c|c|c} \cdot & \cdot & 1 & 1 \\ \cdot & \cdot & 1 & 2 \\ \cdot & \cdot & 2 & 1 \\ \hline \underbrace{\cdot} & \underbrace{\cdot} & \underbrace{2} & \underbrace{2} \\ x_{s,2}^1 & x_{f,2}^1 & x_{l,2}^1(P_{2,1}) & x_{l,2}^1(P_{3,2}) \end{array} \right). \quad (35)$$

X. CONCLUSION

We have presented a new algorithm, DDC- M , for on-line monitoring and diagnosis of modular systems modeled as a set of place-bordered Petri nets. DDC- M exploits the distributed nature of the system to avoid the combinatorial explosion of the state space, but it requires communication among modules on the occurrence of events that affect common places. Many issues remain to be investigated. Among those we mention: further improvements of DDC- M to reduce the communication overhead and deal with communication delays; proper partitioning of a system into modules in order to enhance the performance of DDC- M ; and performance analysis of DDC- M on comprehensive examples using our software tool.

ACKNOWLEDGEMENT

The authors would like to thank the reviewers for their useful comments and suggestions which helped to improve the paper.

REFERENCES

- [1] S. Lafortune, D. Teneketzis, M. Sampath, R. Sengupta, and K. Sinnamohideen, "Failure diagnosis of dynamic systems: An approach based on discrete event systems," in *Proc. 2001 American Control Conf.*, June 2001, pp. 2058–2071.
- [2] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Failure diagnosis using discrete event models," *IEEE Trans. Control Systems Technology*, vol. 4, no. 2, pp. 105–124, Mar. 1996.
- [3] R. Debouk, S. Lafortune, and D. Teneketzis, "Coordinated decentralized protocols for failure diagnosis of discrete-event systems," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 10, no. 1/2, pp. 33–86, Jan. 2000.
- [4] O. Contant, S. Lafortune, and D. Teneketzis, "Diagnosis of modular discrete event systems," in *Proc. of the 2004 International Workshop on Discrete Event Systems - WODES'04*, Reims, France, Sept. 2004.
- [5] J. Sifakis, "Realization of fault-tolerant systems by coding Petri nets," *Journal of Design Automation and Fault-Tolerant Computing* Vol. 3, pp. 93–107, April 1979.
- [6] A. Giua, "Petri net state estimators based on event observation," in *Proc. 36th IEEE Conf. on Decision and Control*, December 1997, pp. 4086–4091.
- [7] C. N. Hadjicostis and G. C. Verghese, "Monitoring Discrete Event Systems Using Petri Net Embeddings," *Application and Theory of Petri Nets 1999 (Series Lecture Notes in Computer Science)*, vol. 1639, pp. 188–207, 1999.
- [8] A. Benveniste, E. Fabre, S. Haar, and C. Jard, "Diagnosis of asynchronous discrete event systems, a net unfolding approach," *IEEE Trans. Automatic Control*, vol. 48, no. 5, pp. 714–727, May 2003.
- [9] R. K. Boel and G. Jiroveanu, "Distributed contextual diagnosis for very large systems," in *Proc. of the 2004 International Workshop on Discrete Event Systems - WODES'04*, Reims, France, Sept. 2004.
- [10] —, "A distributed approach for fault detection and diagnosis based on time Petri nets," in *Proceedings of CESA'03*, Lille, France, July 2003.
- [11] A. Giua, D. Corona, and C. Seatzu, "State estimation of λ -free labeled Petri nets with contact-free nondeterministic transitions," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 15, no. 1, pp. 85–108, Mar. 2005.
- [12] M. H. de Queiroz and J. E. R. Cury, "Modular control of composed systems," in *Proc. 2000 American Control Conf.*, Chicago, USA, June 2000.
- [13] R. Su, W. Wonham, J. Kurien, and X. Koutsoukos, "Distributed diagnosis for qualitative systems," in *Proc. of the 2002 International Workshop on Discrete Event Systems - WODES'02*, M. Silva, A. Giua, and J. Colom, Eds. IEEE Computer Society, Oct. 2002, pp. 169–174.
- [14] R. Su and W. Wonham, "Hierarchical distributed diagnosis under global consistency," in *Proc. of the 2004 International Workshop on Discrete Event Systems - WODES'04*, M. Silva, A. Giua, and J. Colom, Eds., Sept. 2004, pp. 157–162.
- [15] S. Genc and S. Lafortune, "Distributed diagnosis of discrete-event systems using Petri nets," in *Application and Theory of Petri Nets, 2003 (Series Lecture Notes in Computer Science)*, vol. 2679. Springer-Verlag, June 2003, pp. 316–336.

- [16] —, "A distributed algorithm for on-line diagnosis of place-bordered Petri nets," in *16th International Federation of Automatic Control World Congress*, Prague, Czech Republic, July 2005.
- [17] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.
- [18] M. Zhou and F. Dicesare, *Petri net synthesis for discrete event control of manufacturing systems*. Kluwer Academic Publishers, 1993.
- [19] M. Zhou and K. Venkatesh, *Modeling, simulation, and control of flexible manufacturing systems : A Petri net approach*. World Scientific, 1999.
- [20] A. A. Desrochers and R. Y. Al-Jaar, *Applications of Petri nets in automated manufacturing systems : Modeling, control, and performance analysis*. IEEE Press, 1995.
- [21] J.-M. Proth and X. Xie, *Petri nets : A tool for design and management of manufacturing systems*. Wiley, 1996.

APPENDIX

PROOFS OF CORRECTNESS RESULTS

Proof: [of Lemma 1] The proof of the lemma is by construction of DDC- M defined by Algorithms 1 and 2, and induction on the observed sequence of events.

Base ($r = 0$): By construction $x_{l,0}^m(i, P_{m,n}) = x_{l,0}^n(j, P_{m,n}) = \square$ for all rows i and j of $x_{l,0}^m(P_{m,n})$ and $x_{l,0}^n(P_{m,n})$, and $x_{s,0}^m(i_m, P_{m,n}) = x_{s,0}^n(i_n, P_{m,n})$ for any row i_m and i_n .

Hypothesis ($r = R - 1$): Suppose that if $x_{l,R-1}^m(i_m, P_{m,n}) = x_{l,R-1}^n(i_n, P_{m,n})$ for some rows i_m and i_n , then $x_{s,R-1}^m(i_m, P_{m,n}) = x_{s,R-1}^n(i_n, P_{m,n})$.

Step ($r = R$): We show that if $x_{l,R}^m(i_m, P_{m,n}) = x_{l,R}^n(i_n, P_{m,n})$ for some rows i_m and i_n , then $x_{s,R}^m(i_m, P_{m,n}) = x_{s,R}^n(i_n, P_{m,n})$.

If σ_{oR} is neither in $\Sigma_{o,m}$ nor $\Sigma_{o,n}$, then by Algorithm 1, the diagnoser states of the previous iteration $r = R - 1$ stay the same. Thus, the induction step is proved by the induction hypothesis.

If σ_{oR} is either in $\Sigma_{o,m}$ or $\Sigma_{o,n}$, then without loss of generality suppose that $\sigma_{oR} \in \Sigma_{o,m}$. Then, by Line 3 of Algorithm 1 and the definition of the diagnoser state function in Equation (5) we have

$$x_{d,R}^m = \bigcup_{u \in S_m(x_{d,R-1}^m, \sigma_{oR})} UR_m(u). \quad (36)$$

By Equations (6) and (7), for some row $x_{d,R}^m(i_m)$ and $u \in S_m(x_{s,R-1}^m, \sigma_{oR})$,

$$x_{s,R}^m(i_m) = u_s + W_m(\bar{t}_{uo}), \quad (37)$$

where \bar{t}_{uo} is a sequence of unobservable events enabled from u_s .

For all fault types k in $\Delta_{f,m}$, if $u_f(k) = 1$, then $x_{f,R}^m(i_m) = 1$. If $u_f(k) = 0$ and if there exists a transition in the sequence of unobservable events \bar{t}_{uo} which is labeled with an event from the set $\Sigma_{Fk,m}$, then $x_{f,R}^m(i_m) = 1$; otherwise $x_{f,R}^m(i_m) = 0$.

For the message label we have

$$x_{l,R}^m(i_m, P_{m,n}) = u_l(P_{m,n}). \quad (38)$$

Suppose that $u \in S_m(x_{s,R-1}^m, \sigma_{oR})$ is reached from some row $x_{d,R-1}^m(j_m)$ by firing some transition t_o labeled with σ_{oR} . Formally, we have

$$u_s = x_{s,R-1}^m(j_m) + W_m(t_o), \quad (39)$$

$$u_f = x_{f,R-1}^m(j_m), \quad (40)$$

and for all $\mathcal{D}_n \in \mathcal{S}_{\mathcal{D}}$ such that $P_{m,n} \neq \emptyset$, if a message is sent

$$u_l(P_{m,n}) = [x_{l,R-1}^m(j_m, P_{m,n}) W_m(t_o, P_{m,n})], \quad (41)$$

otherwise

$$u_l(P_{m,n}) = x_{l,R-1}^m(j_m, P_{m,n}) \quad (42)$$

as defined by Equation (6) and $t \in B_m(x_{d,R-1}^m, \sigma_{oR})$.

We now consider the two following cases: (1) A message is sent from \mathcal{D}_m to \mathcal{D}_n ; (2) No message is sent.

Case (1) In this case, Equation (41) holds. For all $\mathcal{D}_n \in \mathcal{S}_{\mathcal{D}}$, when a message is received from \mathcal{D}_m , by Line 4 of Algorithm 2 if there exists a row j_m such that $Mesg_{m,n}.Pfx(j_m) = x_{l,R-1}^m(j_n, P_{m,n})$, then by Line 8 of Algorithm 1 $Mesg_{m,n}.Pfx(j_m) = x_{l,R-1}^m(j_m, P_{m,n})$ and by Equation 41, $Mesg_{m,n}.Sfx(j_m) = W_m(t, P_{m,n})$. Thus, there exists rows j_n and j_m such that

$$x_{l,R}^n(j_n, P_{m,n}) = x_{l,R}^m(j_m, P_{m,n}). \quad (43)$$

Then, the diagnoser state $x_{d,R-1}^n(j_n, P_{m,n})$ is updated to $x_{d,R}^n(i_n, P_{m,n})$ by Lines 5, 6 and 7 of Algorithm 2 as follows:

$$x_{s,R}^n(i_n, P_{m,n}) = x_{s,R-1}^n(j_n, P_{m,n}) + W_m(t, P_{m,n}) \quad (44)$$

and

$$x_{s,R}^n(i_n, P_n \setminus P_{m,n}) = x_{s,R-1}^n(j_n, P_n \setminus P_{m,n}), \quad (45)$$

$$x_{l,R}^n(i_n, P_{m,n}) = [x_{l,R-1}^n(j_n, P_{m,n}) W_m(t, P_{m,n})]. \quad (46)$$

By Equation (43) and induction hypothesis $x_{s,R-1}^m(j_m, P_{m,n}) = x_{s,R-1}^n(j_n, P_{m,n})$. Thus, by Equations (39) and (44), $u_s(P_{m,n}) = x_{s,R}^n(i_n, P_{m,n})$. By condition (iii), $W_m(\bar{t}_{uo}, P_{m,n}) = \vec{0}$ in Equation (37), and $x_{s,R}^m(i_m, P_{m,n}) = u_s(P_{m,n}) = x_{s,R}^n(i_n, P_{m,n})$. This completes the proof for *Case (1)*.

Case (2) In this case, Equation (42) holds, and the diagnoser state of \mathcal{D}_n does not change. If $x_{l,R}^m(i_m, P_{m,n}) = x_{l,R}^n(i_n, P_{m,n})$ for some rows i_m and i_n , then by Equation (42), $x_{l,R-1}^m(j_m, P_{m,n}) = x_{l,R-1}^n(j_n, P_{m,n})$ for some rows j_m and j_n and by induction hypothesis, $x_{s,R}^m(j_m, P_{m,n}) = x_{s,R}^n(j_n, P_{m,n})$. If no message is sent, then $W_m(t, P_{m,n}) = \vec{0}$ in Equation (39). Thus, $u_s(P_{m,n}) = x_{s,R-1}^m(j_m, P_{m,n}) = x_{s,R-1}^n(j_n, P_{m,n})$. By condition (iii), $W_m(\bar{t}_{uo}, P_{m,n}) = \vec{0}$ in

Equation (37). Then, $x_{s,R}^m(i_m, P_{m,n}) = u_s(P_{m,n})$. Since the diagnoser state does not change, $x_{s,R-1}^n(j_n, P_{m,n})$ is some row of $x_{d,R}^n$. This completes the proof of *Case (2)* hence the lemma. ■

Proof: [of Lemma 2]

Base (r=0). By construction of the initial diagnoser states $\{x_{d,0}^m : m = 1, 2, \dots, M\}$, $Merge(x_{d,0}^m : \mathcal{D}_m \in \mathcal{S}_{\mathcal{D}}) \neq \emptyset$.

Hypothesis (r=R-1). If the sequence of observable events $\sigma_{o1}\sigma_{o2}\dots\sigma_{oR-1}$ is feasible in $\mathcal{C}_{\mathcal{S}}$, then $Merge(x_{d,R-1}^m : \mathcal{D}_m \in \mathcal{S}_{\mathcal{D}}) \neq \emptyset$.

Step (r=R). If the sequence of observable events $\sigma_{o1}\sigma_{o2}\dots\sigma_{oR}$ is feasible in $\mathcal{C}_{\mathcal{S}}$, then $Merge(x_{d,R}^m : \mathcal{D}_m \in \mathcal{S}_{\mathcal{D}}) \neq \emptyset$.

Proof of Induction Step: Suppose that $\sigma_{o1}\sigma_{o2}\dots\sigma_{oR}$ is a feasible sequence in $\mathcal{C}_{\mathcal{S}_{\mathcal{D}}}$. Then, $\sigma_{o1}\sigma_{o2}\dots\sigma_{oR-1}$ is a feasible sequence. Thus, by the induction hypothesis (since $Merge(x_{d,R-1}^m : \mathcal{D}_m \in \mathcal{S}_{\mathcal{D}}) \neq \emptyset$) $x_{l,R-1}^m(j_m, P_{m,n}) = x_{l,R-1}^n(j_n, P_{m,n})$ for some j_m and j_n , and any module \mathcal{D}_m and \mathcal{D}_n in $\mathcal{S}_{\mathcal{D}}$.

Without loss of generality, we assume that $\sigma_{oR} \in \Sigma_{o,m}$. Since σ_{oR} is enabled in $\mathcal{C}_{\mathcal{S}_{\mathcal{D}}}$, then σ_{oR} is also enabled in the module $\mathcal{D}_m \in \mathcal{S}_{\mathcal{D}}$.

We now differentiate between the two cases: Upon observation of σ_{oR} , (1) a message is sent from \mathcal{D}_m to some module $\mathcal{D}_n \in \mathcal{S}_{\mathcal{D}}$ such that $P_{m,n} \neq \emptyset$, or (2) no message is sent.

Case (1): By the induction hypothesis, Line 4 of Algorithm 2 holds. Thus, $x_{l,R}^m(i_m, P_{m,n}) = x_{l,R}^n(i_n, P_{m,n})$ for some i_m and i_n for all $\mathcal{D}_n \in \mathcal{S}_{\mathcal{D}}$ such that $P_{m,n} \neq \emptyset$.

Case (2): If there is no communication, then $x_{l,R}^m(i_m, P_{m,n}) = x_{l,R-1}^m(j_m, P_{m,n})$ for all $\mathcal{D}_m \in \mathcal{S}_{\mathcal{D}}$. Thus, by induction hypothesis $x_{l,R}^m(i_m, P_{m,n}) = x_{l,R}^n(i_n, P_{m,n})$ for some i_m and i_n for all $\mathcal{D}_m, \mathcal{D}_n \in \mathcal{S}_{\mathcal{D}}$ such that $P_{m,n} = \emptyset$.

By combining Case (1) and (2), and the definition of merge operation, we form $Merge(x_{d,R}^m : \mathcal{D}_m \in \mathcal{S}_{\mathcal{D}}) \neq \emptyset$. ■

Proof: [of Theorem 1] The proof of the theorem is by construction of DDC- M defined by Algorithms 1 and 2, and induction on the observed sequence of events.

Base (r=0). The proof is by construction of $\mathcal{C}_{\mathcal{S}}$ and assumption (iii). By construction $x_0(P_m) = x_0^m$ for any $\mathcal{D}_m \in \mathcal{S}_{\mathcal{D}}$. Suppose we pick some \mathcal{D}_m . Then, by assumption (iii), since the transitions removing tokens from or putting tokens into the common places are labeled with unobservable events, for all $\mathcal{D}_n \in \mathcal{S}_{\mathcal{D}}$ such that \mathcal{D}_m is place-bordered $UR(x_0(P_{m,n})) = x_0(P_{m,n})$. Thus, $UR(x_0(P_m)) = UR(x_0^m)$ and no message label is created. By definition of the diagnoser state transition function in Equation (5), $x_{d,0}$ is the listing of the elements in $UR(x_0(P_m))$. This completes the proof of the base case.

Hypothesis (r=R-1). “Merge($\{x_{d,R-1}^m : \mathcal{D}_m \in \mathcal{S}_{\mathcal{D}}\}$) = $\mathcal{X}_{d,R-1}$.”

Step (r=R). “Merge($\{x_{d,R}^m : \mathcal{D}_m \in \mathcal{S}_{\mathcal{D}}\}$) = $\mathcal{X}_{d,R}$.”

Proof of Induction Step: We show set inclusion of both sides of the equality.

(\subseteq): By Lemma 2, there exists some $y \in \text{Merge}(\{x_{d,R}^m : \mathcal{D}_m \in \mathcal{S}_{\mathcal{D}}\})$ such that

$$y_s(P_m) = x_{s,R}^m(i_m), \quad (47)$$

$$y_f(\Delta_{f,m}) = x_{f,R}^m(i_m) \quad (48)$$

for each $\mathcal{D}_m \in \mathcal{S}_{\mathcal{D}}$.

Without loss of generality we assume that $\sigma_{oR} \in \Sigma_{o,m}$. We differentiate between the two cases: (1) A message is sent from \mathcal{D}_m to \mathcal{D}_n such that $P_{m,n} \neq \emptyset$; (2) No message is sent.

Case (1): If there exists a place-bordered net \mathcal{D}_m such that $P_{m,n} \neq \emptyset$, then there exist some row j_n of the diagnoser state of \mathcal{D}_n such that for some row j_m we have $\text{Mesg}_{m,n}.Pfx(j_m) = x_{l,R-1}^n(j_n, P_{m,n})$, i.e., the condition in Line 4 of Algorithm 2 holds. Since by Line 8 of Algorithm 1 $\text{Mesg}_{m,n}.Pfx(j_m) = x_{l,R-1}^m(j_m, P_{m,n})$, then $x_{l,R-1}^m(j_m, P_{m,n}) = x_{l,R-1}^n(j_n, P_{m,n})$. Then, by induction hypothesis there exists some element $x_{s,R}(j)$ of $\mathcal{X}_{d,R}$ such that

$$x_{s,R-1}(j, P_m) = x_{s,R-1}^m(j_m), \text{ and } x_{s,R-1}(j, P_n) = x_{s,R-1}^n(j_n), \quad (49)$$

$$x_{f,R-1}(j, \Delta_{f,m}) = x_{f,R-1}^m(j_m), \text{ and } x_{f,R-1}(j, \Delta_{f,n}) = x_{f,R-1}^n(j_n). \quad (50)$$

By Equation (49) and Lemma 2, if $t_o \in B_m(x_{d,R-1}^m, \sigma_{oR})$, i.e., t_o is enabled from $x_{s,R-1}^m(j_m)$, then it is also enabled from $x_{s,R-1}(j_m, P_m)$. Similarly, for \bar{t}_{uo} . On the other hand, if we consider the very same Equations (36)-(40) for the place-bordered singleton set $\mathcal{C}_{d,\mathcal{S}}$, then $y \in \mathcal{X}_{d,R}$.

Case (2): Since no message is sent and received, the proof of this case is straightforward by the induction hypothesis.

(\supseteq): Suppose $x_{d,R}(i) \in \mathcal{X}_{d,R}$. Then, there exists $x_{d,R-1}(i) \in \mathcal{X}_{d,R}$ such that the set of Equations (36)-(40) hold when the place-bordered set is the singleton set $\mathcal{C}_{d,\mathcal{S}}$.

By induction hypothesis, there exists $x_{d,R-1}^n(j_n)$ and $x_{d,R-1}^m(j_m)$ such that Equations (49) and (50) hold. Then, we find $x_{d,R}^n(i_n)$ and $x_{d,R}^m(i_m)$ by Equations (36)-(46) such that $x_{d,R}^n(i_n)$ merges with $x_{d,R}^m(i_m)$. Thus,

$$x_{s,R}^m(i_m) = x_{s,R}(i, P_m), \quad (51)$$

$$x_{f,R}^m(i_m) = x_{f,R}(i, \Delta_{f,m}). \quad (52)$$

This completes the proof as $x_{d,R}(i) \in \text{Merge}(\{x_{d,R}^m : \mathcal{M}_{d,m} \in \mathcal{M}_d\})$.

Proof: [of Theorem 2] The proof is similar to the proof of Theorem 1. We follow the very same methodology of the proof of Theorem 1. However, in this proof the message labels and messages have different structures as described by Algorithms 3 and 4. Thus, by Line 6 of Algorithm 3 we rewrite Equation 38 in two steps as follows

$$x_{l,R}^m(i_m, P_{m,n}) = \text{En}(z_{l,R}^m(i_m, P_{m,n})) = \text{En}(u_l(P_{m,n})). \quad (53)$$

By Lines 10 and 11 of Algorithm 3, if $\text{Mesg}_{m,n}.Pfx(j_m) = x_{l,R-1}^m(j_m, P_{m,n})$, then $\text{Mesg}_{m,n}.Sfx(j_m, 1) = W_m(t, P_{m,n})$ and $\text{Mesg}_{m,n}.Sfx(j_m, 2) = x_{l,R}^m(i_m, P_{m,n})$. Thus, Equations (44) and (45) stay the same but by Line 7 of Algorithm 4 Equation (46) becomes

$$x_{l,R}^n(i_n, P_{m,n}) = x_{l,R}^m(i_m, P_{m,n}). \quad (54)$$

These are the only changes in the equations of the proof of Theorem 1 to complete the proof of Theorem .