A DISTRIBUTED ALGORITHM FOR ON-LINE DIAGNOSIS OF PLACE-BORDERED PETRI NETS [1]

Şahika Genç, 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: A new distributed algorithm for on-line fault detection and isolation of discrete-event systems modeled by Petri nets is presented. The algorithm is applicable to systems modeled in a modular manner by means of place-bordered Petri nets, i.e., Petri nets with common places but distinct transitions. These Petri nets have transition labeled with events; fault events are modeled as transitions labeled with unobservable events. It is assumed that the diagnoser modules are able to communicate in real-time during the diagnostic process. 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.

Keywords: Fault diagnosis, distributed algorithms, Petri nets, software implementation.

1. INTRODUCTION

This paper addresses the problem of detecting and isolating faults or other significant events in dynamic systems that can be modeled as a set of interacting discrete-event modules. The events to be detected and isolated (referred to as "faults" hereafter) are modeled as unobservable events in the respective system modules. Fault detection and isolation 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 (Lafortune *et al.*, 2001) and the references therein.

The methodology termed the "Diagnoser Approach", introduced in (Sampath *et al.*, 1996) and subsequently extended in several works including (Debouk *et al.*, 2000), 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 (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. Recently, a diagnoser-based methodology for on-line monitoring of discrete-event systems modeled by Petri nets was proposed in (Genc and Lafortune, 2003). As in (Sampath *et al.*, 1996), diagnosers are used for on-line monitoring; however, Petri nets - not automata - are used to represent the transition structures of these diagnosers. The use of Petri nets offers potential advantages over automata such as compactness of the transition structure and distributed nature of the state.

This paper builds on and extends the results in (Genc and Lafortune, 2003). The work in (Genc and Lafortune, 2003) deals with systems described by (i) monolithic models, leading to *centralized* Petri net diag-

[1] 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.

nosers, and (ii) models composed of *two* Petri nets sharing a set of common places, leading to a distributed diagnosis algorithm with communication abbreviated as "DDC-2" hereafter.[2] In this paper, we consider the case of modular systems consisting of a *set* of place-bordered Petri nets and present a new algorithm that extends DDC-2 to the case of multiple modules. The extension of DDC-2 to multiple modules and in particular the associated correctness proof of the new algorithm are non-trivial extensions of the results in (Genc and Lafortune, 2003).

Petri net models have been employed to solve problems of state observability, system monitoring, alarm analysis, and fault diagnosis in several works, including (Sifakis, 1979; Giua, 1997; Hadjicostis and Verghese, 1999; Benveniste *et al.*, 2003; Boel and Jiroveanu, 2004). However, to the best of our knowledge, reference (Genc and Lafortune, 2003) and this paper are the first to explore the extension of the Diagnoser Approach of (Sampath *et al.*, 1996) to Petri net models. 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., (de Queiroz and Cury, 2000; Su *et al.*, 2002; Benveniste *et al.*, 2003; Contant *et al.*, 2004). The suitability of Petri nets to model distributed systems was a key motivation for the use of Petri net structures in the work in (Benveniste *et al.*, 2003) 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.

The remainder of this paper is organized as follows. We present in Section 2 our new distributed algorithm with communication, abbreviated DDC-*M*, for diagnosing systems composed of M modules, $M \geq 2$. Results about the correctness of the DDC-*M* are presented in Section 3. In Section 4, we briefly present the software implementation of DDC-*M* in MATLAB. Finally, some concluding remarks are given in Section 5.

2. DISTRIBUTED DIAGNOSIS WITH COMMUNICATION

In this section, we study the problem of distributed diagnosis with communication. We are interested in the case where the system consists of a set of modules and diagnosers are constructed for each module. We allow these diagnosers to communicate.

We start with some general definitions. 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 $w : A \to \mathbb{Z}_+$ is the weight function on the arcs. We denote by $W(P,t)$ the 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, $l : \Sigma \to T$ 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 $f : X \times T \to X$ gives the resulting state.

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.

2.1 System Model

The system to be diagnosed is given by a set of place-bordered labeled Petri nets $\mathscr{M} = \{(\mathcal{N}_m, \Sigma_m, l_m, x_0^m) : m = 1,2,\ldots,M\}$, where the additional symbol m identifies the *module* in the set and M is the number of modules. The set of fault events of type k of module $\mathscr{M}_m$ is denoted by $\Sigma_{Fk,m}$. We assume the following conditions $\forall \mathscr{M}_m \in \mathscr{M}$: (i) $\forall \mathscr{M}_n \in \mathscr{M}$, $\Sigma_m \cap \Sigma_n = \emptyset$, (ii) $\exists \mathscr{M}_n \in \mathscr{M}$, such that $P_{m,n} = P_m \cap P_n \neq \emptyset$, (iii) $\forall t \in T_m$ if t puts tokens into or removes tokens from $P_{m,n}$ for some $\mathscr{M}_n \in \mathscr{M}$, then $l_m(t) \in \Sigma_{o,m}$. Thus, the modules in $\mathscr{M}$ have disjoint sets of transitions but share at least one place with another module in the set. 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.

Example 1. Consider the valve and load model in Fig. 1, controller model in Fig. 2 and a pump model which is graphically isomorphic to the valve up to renaming of events and places. In the pump, the events of the valve are renamed as follows: *valve_open* to *start_pump*, *close_valve* to *stop_pump*, *stuck_closed* to *pump_failed_off*, *stuck_open* to *pump_failed_on*; places are enumerated with prefix *pm* instead of *vl*. We take the parallel composition of the valve, pump, load and controller (not shown here). The valve, pump and load in Figs. 3, 4, 5 form the set of place-bordered nets that constitute the overall system model. The places of the controller in Fig. 2 are the common places between these place-bordered nets. Figure 6 shows the interconnection between the individual place-bordered nets. For all the nets in this paper, the filled transi-

[2] DDC-2 is denoted by DDC in (Genc and Lafortune, 2003); the "-2" label has been added in this paper for the sake of clarity.

tions are labeled with unobservable events. The fault types are: $\Sigma_{F1,1} = \{stuck\_open\_1,\ stuck\_open\_2\}$, $\Sigma_{F2,1} = \{stuck\_closed\_1,\ stuck\_closed\_2\}$, $\Sigma_{F1,2} = \{pump\_\ failed\_on\_1,\ pump\_failed\_on\_2\}$, $\Sigma_{F2,2} = \{pump\_failed\_off\_1, pump\_failed\_off\_2\}$, $\Sigma_{F1,3} = \{failed\_off\}$.

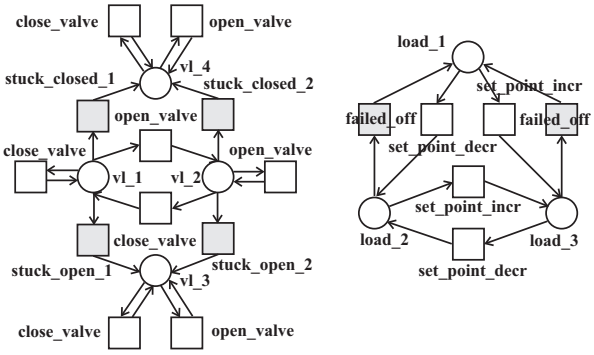

Fig. 1. The valve and load model.

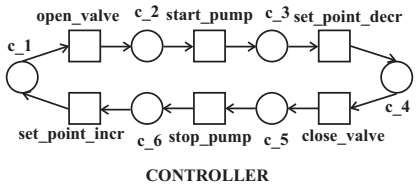

Fig. 2. The controller model.

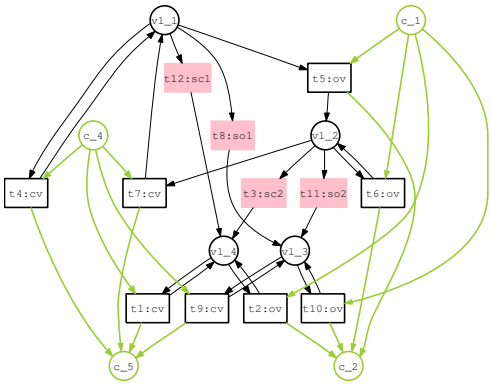

Fig. 3. Place-bordered net: Module#1.

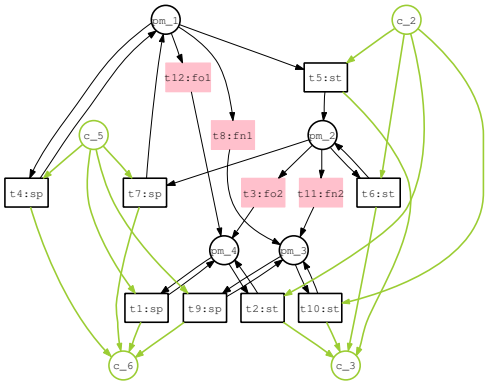

Fig. 4. Place-bordered net: Module#2.

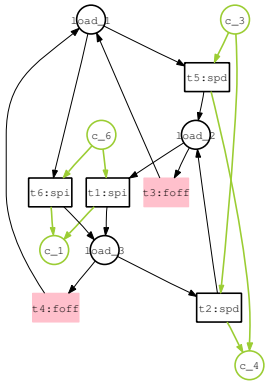

Fig. 5. Place-bordered net: Module#3.

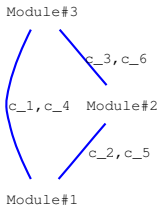

Fig. 6. Common places between the modules.

2.2 *Communicating Petri Net Diagnosers*

2.2.1. Petri Net Diagnosers We construct a Petri net diagnoser for each module in the set $\mathscr{M}$ and denote the resulting set of diagnosers by $\mathscr{M}_d = \{(\mathscr{N}_m, \Sigma_m, l_m, x_0^{d,m}, \Delta_{f,m}) : m = 1, 2, \ldots, M\}$, where $\Delta_{f,m}$ is the set of fault types of $\mathscr{M}_{d,m}$. Petri net diagnosers were first defined in (Genc and Lafortune, 2003). Due to space limitations, some familiarity with (Genc and Lafortune, 2003) is assumed in the sequel. Our Petri net diagnosers differ from those in (Genc and Lafortune, 2003) in terms of structure of message labels. We present the salient features of these diagnosers.

The diagnoser state x_d^m of module $\mathscr{M}_{d,m} \in \mathscr{M}_d$ is a matrix of the form

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

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 $\mathscr{M}_m$ following the occurrence of the observed sequence of events.

The diagnoser state transition function of $\mathscr{M}_{d,m} \in \mathscr{M}_d$ is $f_{d,m} : X_d^m \times \Sigma_{o,m} \to X_d^m$, where X_d^m is the state space of $\mathscr{M}_{d,m}$. 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

$$\cup_{u \in S_m(x_d^m, a)} UR_m(u), \qquad (1)$$

where $S_m(x_d^m, a)$ 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 and $UR_m(u)$ 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

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

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 $a \in \Sigma_{o,m}$. The definition of unobservable reach of a (place-bordered Petri net) diagnoser state is given in (Genc and Lafortune, 2003) and omitted here. Fault labels are used as in automata diagnosers to memorize the occurrence of a fault event in the diagnoser state; see (Genc and Lafortune, 2003) for further details on $x_f^m(i)$. Examination of the current fault type reveals the status of the different types of faults: fault(s) of Type Fk did not occur, fault(s) of Type Fk possibly occurred ("Fk-uncertain state"), fault(s) of Type Fk occurred for sure ("Fk-certain state").

Finally, the $x_l^m(i)$ part of each row of the diagnoser state corresponds to the message label. Message labels memorize the history of token additions/removals for common places. For convenience, we divide the message label into different parts where each part pertains to common places (if any) between two given modules. 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 $\mathscr{M}_n \in \mathscr{M}$ 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 $y_l^m(j, P_{m,n}).Sfx = W(P_{m,n}, t)$.

The module and corresponding diagnoser have the same Petri net graph. However, the modules do not have disjoint sets of places and can affect 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 their state estimates will be incomplete. We overcome this problem by defining a communication protocol between diagnosers, presented next.

2.2.2. Communication Protocol We now formalize our DDC-*M* algorithm for distributed diagnosis of communicating Petri net diagnosers. DDC-*M* is presented in two parts, called Algorithms 1 and 2, respectively. 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: The outer loop (line 1) considers for the sake of generality a sequence of observable events. Consider these events one at a time. 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, 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).[3] The effect 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 the 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

[3] We have developed techniques for preventing unbounded growth of message labels; these are not presented here due to space constraints.

is the value returned by the function *UDSC*. Note that it is not necessary to perform the unobservable reach since we assume that transitions out of common places are labeled by observable events.

Algorithm 1 Distributed Diagnosis with Communication

Require: The sequence $\sigma_{o1}\sigma_{o2}\dots\sigma_{oR}$ is observed.
for $r = 1 : R$ **do**
  Find $\mathscr{M}_m$ such that $\sigma_{or} \in \Sigma_m$,
  $x^m_{d,r} \leftarrow f_{m,d}(x^m_{d,r-1}, \sigma_{or})$.
  for all $\mathscr{M}_{d,n} \in \mathscr{M}_d$ such that $P_{m,n} \neq \emptyset$ **do**
    if $\{W(P_{m,n},t) | t \in B_m(x^m_{d,r-1}, \sigma_{or})\} \neq \{0\}$ **then**
      $Mesg_{m,n} \leftarrow \{\ \}$,
      for all j=# of rows of $x^m_{l,r}$ **do**
        $Mesg_{m,n}.Pfx(j) \leftarrow x_{l,r}(j, P_{m,n}).Pfx$,
        $Mesg_{m,n}.Sfx(j) \leftarrow x_{l,r}(j, P_{m,n}).Sfx$,
        $Mesg_{m,n}(j) \leftarrow (Mesg_{m,n}.Pfx(j), Mesg_{m,n}.Sfx(j))$,
      end for
      Send all different rows of $Mesg_{m,n}$,
      $x^n_{d,r} \leftarrow UDSC(x^n_{d,r-1}, Mesg_{m,n})$,
    end if
  end for
end for

Algorithm 2 Update of Diagnoser State upon Communication

Require: $x^n_{d,r-1}, Mesg_{m,n}$
$X^n_{d,r} \leftarrow \{\ \}$,
for all $i = 1$: Number of rows of $Mesg_{m,n}.Pfx$ **do**
  for all $j = 1$: Number of rows of $x^n_l(P_c(n,m))$ **do**
    if $Mesg_{m,n}.Pfx(i) == x^n_l(j, P_c(n,m))$ **then**
      $y_s(P_{m,n}) := x^n_{s,r-1}(j, P_{m,n}) + Mesg_{m,n}.Sfx(i)$
      $y_s(P(n) \setminus P_{m,n}) \leftarrow x^n_s(j, P(n) \setminus P_{m,n})$
      $y_f \leftarrow x^n_f(j)$
      $y_l(P_c(n,m)) \leftarrow (x^n_{l,r-1}(j, P_c(n,m)),\ Mesg_{m,n}.Sfx(i))$
      for all $\mathscr{M}_{d,q} \in (\mathscr{M}_d \setminus \mathscr{M}_{d,m})$ such that $P_c(n,q) \neq \emptyset$ **do**
        $y_l(P_c(n,q)) \leftarrow x^n_{l,r-1}(j, P_c(n,q))$
      end for
      $X^n_d \leftarrow X^n_d \cup [y_s | y_f | y_l]$
    end if
  end for
end for
$UDSC(x^n_{d,r-1}, Mesg_{m,n}) \leftarrow$ Listing of the set $X^n_{d,r}$

Example 2. In this example we illustrate the application of DDC-*M*. Consider the set of place-bordered nets of Example 1. Suppose that initially there is only one token at each of the following places: c_1, vl_1, pm_1 and $load_1$. Consider the following ordered lists of places and the common places for states and message labels: $P_1 = \{c_1, c_2, c_4, c_5, vl_1, vl_2, vl_3, vl_4\}$, $P_2 = \{c_2, c_3, c_5, c_6, pm_1,\ pm_2, pm_3, pm_4\}$, $P_3 = \{c_1, c_3, c_4, c_6, load_1, load_2,\ load_3\}$, $P_{1,2} = P_{2,1} = \{c_2, c_5\}$, $P_{1,3} = P_{3,1} = \{c_1, c_4\}$ and $P_{2,3} = P_{3,2} = \{c_3, c_6\}$. In the message label, neighbor modules are ordered in ascending module numbers.

Then, the initial diagnoser states of the modules are as follows: $x^1_{d,0} = [10001000|00; 10000010|10; 10000001|01]$, $x^2_{d,0} = [00001000|00; 00000010|10; 00000001|01]$, and $x^3_{d,0} = [1000100|0]$. 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 state, fault information, and message labels (consistent with the ordering of the rows) of the diagnoser states obtained by Algorithms 1 and 2 are as follows. Valve: $x^1_{d,1} = [01000001|01|10 : -10;\ 01000010|10|10 : -10;\ 01000100|00|10 : -10]$. Pump: $x^2_{d,2} = [10000001|01|10 : ;\ 10000010|10|10 : ;\ 10001000|00|10 :]$. Load: $x^3_{d,3} = [0000100|0| - 10 :]$. 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*, the state, fault information, and message labels (consistent with the ordering of the rows) of the new diagnoser states are as follows. Valve: $x^1_{d,1} = [00000001|01|10 - 10 : -10;\ 00000010|10|10 - 10 : -10;\ 00000100|00|10 - 10 : -10]$. Pump: $x^2_{d,2} = [01000001|01|10 - 10 : 10;\ 01000010|10|10 - 10 : 10;\ 01000100|00|10 - 10 : 10]$. Load: $x^3_{d,3} = [0100100|0| - 10 : 10]$. 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) $x^1_{f,0}$, $x^1_{f,1}$ and $x^1_{f,2}$ 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) $x^2_{f,0}$, $x^2_{f,1}$ and $x^2_{f,2}$ are both $F_{1,2} - uncertain$ and $F_{2,2} - uncertain$, and (iii) $x^3_{f,0}$, $x^3_{f,1}$ and $x^3_{f,2}$ are normal.

3. RESULTS

We define an operation called *merge* that combines the diagnoser states of the modules.

Definition 3. (Merge). Given the set of place-bordered nets $\mathscr{M}$, and the set of corresponding diagnosers $\mathscr{M}_d$, let $\{x^m_d : m = 0, 1, 2, \dots, M\}$ be the set of diagnoser states of the modules $\mathscr{M}_{d,m} \in \mathscr{M}_d$ after some sequence of observable events. We define the merge operation on these states recursively as follows:

(1) Merge of two diagnoser states, $\mathscr{M}_{d,m}, \mathscr{M}_{d,n} \in \mathscr{M}_d$. There are two cases:
  (a) $P_{m,n} = \emptyset$. In this case for all rows i_m, i_n of x^m_d and x^n_d, respectively,
  $$(x^m_s(i_m, P_m), x^n_s(i_n, P_n) \mid x^m_f x^n_f) \in Merge(x^m_d, x^n_d)(P_m \cup P_n \mid \Delta_{f,m} \cup \Delta_{f,n}).$$
  (b) $P_{m,n} \neq \emptyset$. In this case for all rows i_m, i_n of x^m_d and x^n_d, respectively, such that $x^m_l(i_m, P_{m,n}) = x^n_l(i_n, P_{n,m})$,
  $$(x^m_s(i_m, P_m), x^n_s(i_n, P_n \setminus P_m) \mid x^m_f x^n_f) \in Merge(x^m_d, x^n_d)(P_m \cup P_n \mid \Delta_{f,m} \cup \Delta_{f,n}).$$

(2) Let $\mathscr{M}_{d,m}, \mathscr{M}_{d,n}, \mathscr{M}_{d,q} \in \mathscr{M}_d$. Then,
$$Merge(x^m_d, x^n_d, x^q_d) = Merge(Merge(x^m_d, x^n_d), x^q_d).$$

The intuition behind the merge of two diagnoser states for modules with common places is that composed states are formed 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)).

Theorem 4. DDC-*M* is correct in the sense that the merge operation recovers the corresponding monolithic diagnoser state. [4]

4. SOFTWARE IMPLEMENTATION OF DDC-*M*

We developed a software implementation of DDC-*M* and of the *merge* operation [5]. All the analysis results of the examples in this paper were performed using the software tool.

The software interacts with *GraphViz* developed by AT&T to plot the labeled Petri nets and show the diagnoser states. Petri nets are loaded by their incidence matrices using either graphical tools or user created files. Users can also compose several Petri nets with a controller as was done for the example in this paper using a graphical interface. The software exploits MATLAB's matrix manipulation functions and search algorithms together with internal data types such as structure and cell arrays in order to efficiently implement the *for-loops* in Algorithms 1 and 2.

Our plan is to use the software tool on comprehensive Petri net examples in order to get further insight into the performance of DDC-*M*.

5. CONCLUSION

We have presented a new algorithm, DDC-*M*, for online 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: tuning of DDC-*M* to reduce the size of messages (labels) 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. Regarding the reduction of the size of the message labels, we make three observations. First, message labels can be reset to "null" each time the merge operation is performed. Second, it is possible to shorten message labels by truncating prefixes that are the same for all rows. Third, encoding techniques that keep the message labels at a fixed-size are being developed.

[4] The proof of this result is available at http://www.eecs.umich.edu/∼sgenc/ifac05/proof_thm.pdf.

[5] The reader is referred to http://www.eecs.umich.edu/∼sgenc/ifac05.html.

REFERENCES

Benveniste, A., E. Fabre, S. Haar and C. Jard (2003). Diagnosis of asynchronous discrete event systems, a net unfolding approach. *IEEE Trans. Automatic Control* **48**(5), 714–727.

Boel, R. K. and G. Jiroveanu (2004). Distributed contextual diagnosis for very large systems. In: *Proc. of the 2004 International Workshop on Discrete Event Systems - WODES'04*. Reims, France.

Contant, O., S. Lafortune and D. Teneketzis (2004). Diagnosis of modular discrete event systems. In: *Proc. of the 2004 International Workshop on Discrete Event Systems - WODES'04*. Reims, France.

de Queiroz, M. H. and J. E. R. Cury (2000). Modular control of composed systems. In: *Proc. 2000 American Control Conf.*. Chicago, USA.

Debouk, R., S. Lafortune and D. Teneketzis (2000). Coordinated decentralized protocols for failure diagnosis of discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications* **10**(1/2), 33–86.

Genc, S. and S. Lafortune (2003). 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. pp. 316–336.

Giua, Alessandro (1997). Petri net state estimators based on event observation. *IEEE 36th Int. Conf. on Decision and Control* pp. 4086–4091.

Hadjicostis, Christoforos N. and George C. Verghese (1999). Monitoring discrete event systems using Petri net embeddings. *Application and Theory of Petri Nets 1999 (Series Lecture Notes in Computer Science)* **1639**, 188–207.

Lafortune, S., D. Teneketzis, M. Sampath, R. Sengupta and K. Sinnamohideen (2001). Failure diagnosis of dynamic systems: An approach based on discrete event systems. In: *Proc. 2001 American Control Conf.*. pp. 2058–2071.

Sampath, M., R. Sengupta, S. Lafortune, K. Sinnamohideen and D. Teneketzis (1996). Failure diagnosis using discrete event models. *IEEE Trans. Control Systems Technology* **4**(2), 105–124.

Sifakis, Joseph (1979). Realization of fault-tolerant systems by coding Petri nets. *Journal of Design Automation and Fault-Tolerant Computing Vol. 3* pp. 93–107.

Su, R., W.M. Wonham, J. Kurien and X. Koutsoukos (2002). Distributed diagnosis for qualitative systems. In: *Proc. of the 2002 International Workshop on Discrete Event Systems - WODES'02* (M. Silva, A. Giua and J.M. Colom, Eds.). IEEE Computer Society. pp. 169–174.