International Journal on Artificial Intelligence Tools
© World Scientific Publishing Company

A DECENTRALIZED MODEL-BASED DIAGNOSTIC TOOL FOR COMPLEX SYSTEMS

Y. PENCOLÉ, M-O. CORDIER and L. ROZÉ.

IRISA, University of Rennes 1, campus de Beaulieu, 35000 Rennes, France
ypencole@irisa.fr

IRISA, University of Rennes 1, campus de Beaulieu, 35000 Rennes, France
cordier@irisa.fr

IRISA, INSA, campus de Beaulieu, 35000 Rennes, France
roze@irisa.fr

Received (received date)
Revised (revised date)

We address the problem of diagnosing complex discrete-event systems such as telecommunication networks. Given a flow of observations from the system, the goal is to explain those observations by identifying and localizing possible faults. Several model-based diagnosis approaches deal with this problem but they need the computation of a global model which is not feasible for complex systems like telecommunication networks. Our contribution is the proposal of a decentralized approach which permits to carry out an on-line diagnosis without computing the global model. This paper describes the implementation of a tool based on this approach. Given a decentralized model of the system and a flow of observations, the program analyzes the flow and computes the diagnosis in a decentralized way. The impact of the merging strategy on the global efficiency is demonstrated and illustrated by experimental results on a real application.

Keywords: Model-based diagnosis, discrete event systems, telecommunication networks, monitoring

1. Introduction

The problem we deal with is the supervision of complex and large discrete-event systems such as telecommunication networks. Given a supervisor continuously receiving observations (alarms) sent by the system components, our purpose is to help operators to identify failures. Two classical approaches in monitoring such systems are knowledge-based techniques that directly associate a diagnosis to a set of symptoms, for example expert systems,[1] or chronicle recognition systems,[2,3] and model-based techniques which rely on a behavioral model of the system.[4] The main weakness of the first approach is the lack of genericity: as the network changes, a new expertise has to be acquired. We therefore decided to use model-based techniques which are recognized to be more adapted to evolutive systems than expertise-based approaches are and we focus in the following on these ones.

In the literature, more and more approaches are proposed for diagnosing discrete-event systems, in both AI and control engineering domain. It concerns continuous-variable systems which, after quantization, are represented as discrete systems,[5] as well as "discrete by nature" systems such as communicating processes which exchange messages and alarms. The majority of these approaches are centralized approaches. [6,5,7] The diagnoser approach consists in the compilation of diagnostic information in a data structure (called *diagnoser*) which maps failures and observations for on-line diagnosis. [8] As telecommunication networks are concerned, an extension of this approach has been proposed, which is well-adapted for on-line diagnosis.[7,9] Nevertheless, a centralized approach needs to have a global information about the system which is unrealistic for complex and large systems like telecommunication networks. Moreover, such systems are naturally distributed so it is easier to model those systems in a decentralized way. An approach for diagnosing discrete-event systems using decentralized diagnosers has been proposed, [10] but the computation of each decentralized diagnoser is based on a global model which is a major drawback of this method. On an other hand,there exist methods relying on a decentralized model, [11,12] but these methods are used *off-line* to solve a diagnosis problem *a posteriori*.

Our contribution is to propose an approach which relies on a decentralized model and provides on-line diagnosis of large discrete-event systems such as telecommunication networks. The idea is to split the flow of observations into temporal windows. For each temporal window, we compute a diagnosis for each component of the system (*local diagnosis*) and then we build a diagnosis of the whole system (*global diagnosis*) by merging these local diagnoses and the diagnosis of previous temporal windows. The challenge is to preserve the efficiency of the global computation which depends mainly on the merging operation of local diagnoses. [13] In this paper, we consider the diagnostic task in a given temporal window and focus on the merging operation. The way successive temporal windows are incrementally taken into account is the subject of another paper.[14]

In this paper, we present a tool which implements this approach and we analyse some experimental results on a real application : a French packet-switching network. The paper is organized as follows. We first introduce what kind of systems are considered and the problem of monitoring. Then, we present the formalism based on communicating automata, which is used to represent in a decentralized way the model of the system. Then, the diagnostic task is explained by defining observations and diagnoses. Then, we focus on the merging operation which allows to build step by step the global diagnosis and show the importance of the merging strategy. We then present the application and the diagnostic tool which implements the approach. Experimental results are given which demonstrate the profit taken from the use of an adequate merging strategy.

2. Decentralized model of the system

2.1. *The monitoring task*

The systems we consider are distributed systems composed of *components* which interact each other (see figure 1). Typical components are switches, multiplexers, control stations... Each component can emit some *observations* (also called *alarms*) when a problem occurs on the network. For most of the faults, there are some automatic recovery procedures so that faults are in general not permanent. A *supervision center* is in charge of monitoring the network by analyzing the received alarms in order to determine if an intervention is needed or not. The main problem is the important number of received alarms (thousands of alarms) which makes the monitoring task difficult without the help of automatic diagnostic methods.

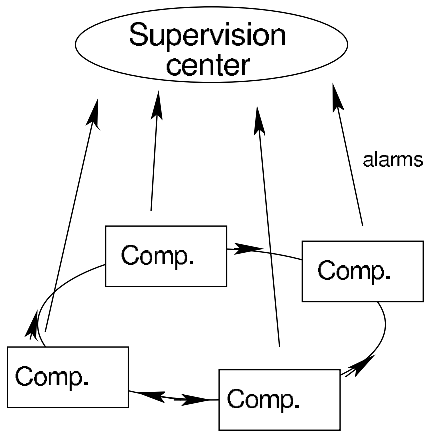

Fig. 1. Supervision of a telecommunication network.

2.2. *Modeling the system*

As said in the introduction, we decided to use model-based approaches which are recognized to be more adapted for evolutive systems as telecommunication networks are. Due to the great number of components, it is quite unrealistic to rely on a global model of such systems. This section explains how the model of the system is described in a decentralized way by means of local models, which describe the behaviors of each component of the system and the interactions between them. The formalism chosen for such models is that of communicating automata.

Each component reacts to failure events by changing of states and emitting observable events to the supervisor. The components interact by exchanging messages (named *internal events*) which occur for instance when failures propagate through the system. The supervisor receives alarms from each component of the system via a communication channel. Each alarm is labeled by the component which sent it. We suppose in the following that the communication channels are FIFO data structures. However, communication channels from different components are not synchronized (their propagation delays can be different). Consequently, we have :

Hypothesis 1 : *Let o_1 and o_2 be observable events emitted by one component. We assume that o_1 and o_2 are received by the supervisor in the order of their emission by the component.*

Hypothesis 2 : *Let o_1 and o_2 be observable events emitted by two different components. We assume that o_1 and o_2 may not be received by the supervisor in the order of their emission by the components.*

A *component* is faced to two kinds of received events: failure events (Σ^i_{fail}) and internal events ($\Sigma^i_{intreceived}$). A component emits two kinds of events: observable events (Σ^i_{obs}) via its communication channel and internal events ($\Sigma^i_{intemitted}$).

Definition 1 (Model of a component) *A* component *behavior is described by a communicating finite-state machine* $\Gamma_i = (\Sigma^i_{in}, 2^{(\Sigma^i_{out})}, Q_i, E_i)$ *where*

- Σ^i_{in} *is the set of input events* ($\Sigma^i_{in} = \Sigma^i_{fail} \cup \Sigma^i_{intreceived}$)*;*
- Σ^i_{out} *is the set of output events* ($\Sigma^i_{out} = \Sigma^i_{obs} \cup \Sigma^i_{intemitted}$)*;*
- Q_i *is the set of states of the component;*
- $E_i \subseteq (Q_i \times \Sigma^i_{in} \times 2^{(\Sigma^i_{out})} \times Q_i)$ *is the set of transitions.*

A model of component is depicted on Figure 2. Two kinds of failure events can happen on this component: *f1* and *f2*. For example, if *f1* occurs on the component at state 1, then the component emits the *a1* alarm, propagates the failure by the emission of the *i2* event towards another component and goes to state 2.

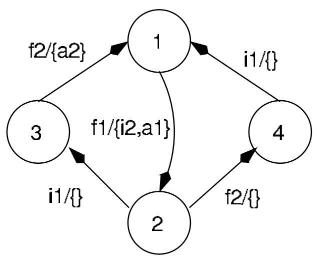

Fig. 2. A model of component: *f1* and *f2* are failure events, *i1* and *i2* are events received from/emitted towards another component, *a1* and *a2* are alarms.

The model of the system is described in a decentralized way by the models of its components.

Definition 2 (Model of a system) *The* model Γ *of a system is given by the set of models of its components* $\{\Gamma_1, \ldots, \Gamma_n\}$*, a set of failure events* (Σ_{fail})*, a set of observable events* (Σ_{obs}) *and a set of internal events* (Σ_{int}) *such that:*

- $\{\Sigma^1_{obs}, \ldots, \Sigma^n_{obs}\}$ *is a partition of* Σ_{obs};
- $\{\Sigma^1_{fail}, \ldots, \Sigma^n_{fail}\}$ *is a partition of* Σ_{fail};
- $\{\Sigma^1_{intreceived}, \ldots, \Sigma^n_{intreceived}\}$ *and* $\{\Sigma^1_{intemitted}, \ldots, \Sigma^n_{intemitted}\}$ *are partitions of* Σ_{int};
- $\forall e \in \Sigma_{int}, \exists! \Gamma_i \mid e \in \Sigma^i_{intreceived} \wedge \exists! \Gamma_j \mid e \in \Sigma^j_{intemitted} \wedge i \neq j$.

We suppose that the system reacts to one exogenous event at the same time (two exogenous events cannot occur at the same time). This reaction is represented in the model by the triggering of a set of transitions from different models of components. The occurrence of an exogenous event in the system is represented by the triggering of a transition labeled with this event. If this transition emits internal events, it activates the triggering of transitions labeled as input with such internal events: such activations represent the propagation of the failure. The global model of the system which expresses this behavior, could be explicitly built by composing the automata of its components (via a classical operation of synchronization on the internal events, [15]) but it is exactly what we want to avoid due to the important size of such a model for large systems.

3. Decentralized diagnosis task

The diagnostic task consists of inferring information on the faulty state of the system components and/or on the occurrence of faulty events from the sequence of observations (in our case the sequence of alarms received by the supervisor). As many authors, [11,16,17], we consider that the diagnostic task consists, for dynamical systems, in finding the trajectories (sequences of states and transitions) explaining the observations. From these trajectories, it is in general easy to extract the diagnosis itself. A simple case is when you are only interested in the set of possible faulty components but in a monitoring context, the operator is often interested in getting a more thorough explanation of what happened, as at least the sequences of exogenous failure events explaining the set of received alarms and the possible current states of the system.

The idea is to use a decentralized diagnosis approach by firstly computing a diagnosis for each component (*local diagnosis*) and then building a diagnosis of the whole system (*global diagnosis*) from these local diagnoses. In the following, we formally define these notions and explain how the global diagnosis is computed by *composing* the local ones.

In the following, the set of observation sequences is noted : $\mathcal{O} = \{\mathcal{O}_{\Gamma_1}, \ldots, \mathcal{O}_{\Gamma_n}\}$ with $\mathcal{O}_{\Gamma_i} \in (\Sigma^i_{obs})^\star$. Each $\mathcal{O}_{\Gamma_i}$ is the sequence of observations received from the component Γ_i during a temporal window Δ_t[1]. At the beginning of the temporal window, the initial states X^{Γ}_{init} of Γ are known and described by the initial states of each component $\{X^{\Gamma_1}_{init}, \ldots, X^{\Gamma_n}_{init}\}$.

[1]As said in the introduction, we consider the diagnotic task in a given temporal window. The way successive temporal windows are incrementally taken into account the subject of another paper.[14]

3.1. *Local diagnosis*

Given the model of the component Γ_i and a set of initial states $X^{\Gamma_i}_{init}$, a *local diagnosis* Δ_{Γ_i} describes the subset of trajectories from Γ_i starting from elements of $X^{\Gamma_i}_{init}$ and explaining the sequence of *local* observations $\mathcal{O}_{\Gamma_i}$. In other words, if the diagnosed component follows one of those trajectories then the sequence of emitted observable events according to the model Γ_i is exactly $\mathcal{O}_{\Gamma_i}$.

So we propose to represent a local diagnosis as a communicating finite-state machine $\Delta_{\Gamma_i}(X^{\Gamma_i}_{init}, \mathcal{O}_{\Gamma_i})$, shortly Δ_{Γ_i}. Compared to the automaton Γ_i, the main syntactical d-ifference is that each state Q_{Δ_i} of this automaton is associated to a pair $(s_{\Gamma_i}, \mathcal{E}_i)$ where $s_{\Gamma_i} \in Q_i$ is a state of the component and $\mathcal{E}_i$ is the prefix subsequence of $\mathcal{O}_{\Gamma_i}$ explained in this state (see figure 3).

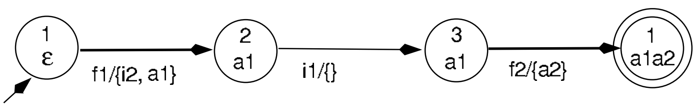

Fig. 3. A local diagnosis of the component presented in figure 2. This diagnosis explains the sequence of alarms *a1a2* from the state 1. It contains one trajectory.

The initial states of Δ_{Γ_i} are those corresponding to $X^{\Gamma_i}_{init}$, in other words, the initial states are $(s_{\Gamma_i}, \mathcal{E}_i)$ where $s_{\Gamma_i} \in X^{\Gamma_i}_{init}$ and $\mathcal{E}_i = \epsilon$. The final states are those such that $\mathcal{E}_i = \mathcal{O}_{\Gamma_i}$. So the final states represent the current states of the component explaining the whole sequence of local observations. As in Γ_i, the transitions are labeled with failures or internal received events as input and with observed or internal emitted events as output. These internal events describes the *interactions* between the component Γ_i and the other components of the system. The local diagnosis of Γ_i describes the trajectories of Γ_i which explains the local observations $\mathcal{O}_{\Gamma_i}$ *under the hypothesis that these interactions can exist.* This hypothesis which correspond to the synchronization of the components will be checked by the merging operation described below.

Definition 3 (Local diagnosis) *The local diagnosis* $\Delta(X^{\Gamma_i}_{init}, \mathcal{O}_{\Gamma_i})$ *of* Γ_i *according to the sequence* $\mathcal{O}_{\Gamma_i}$ *is a finite-state machine:* $(Q_{\Delta_i}, \Sigma^i_{in}, \Sigma^i_{out}, I_{\Delta_i}, F_{\Delta_i}, E_{\Delta_i})$ *where*

- Σ^i_{in} *is the set of input events* ($\Sigma^i_{in} = \Sigma^i_{fail} \cup \Sigma^i_{intreceived}$)*;*
- Σ^i_{out} *is the set of output events* ($\Sigma^i_{out} = \Sigma^i_{obs} \cup \Sigma^i_{intemitted}$)*;*
- Q_{Δ_i} *is the set of states* $(Q_{\Delta i} \subseteq Q_i \times (\Sigma^i_{obs})^\star)$*;*
- $I_{\Delta_i} \subseteq Q_{\Delta_i}$ *is the set of initial states;*
- $F_{\Delta_i} \subseteq Q_{\Delta_i}$ *is the set of final states;*
- $E_{\Delta_i} \subseteq (Q_{\Delta_i} \times 2^{\Sigma^i_{in}} \times 2^{\Sigma^i_{out}} \times Q_{\Delta_i})$ *is the set of transitions.*

Thus, a local diagnosis Δ_{Γ_i} is computed, given the local model Γ_i, a set of initial states X^i_{init} of Γ_i and a sequence of local observations. There exist several algorithms for computing such a diagnosis. Basically, they are based on a transition path search algorithm (depth-first search, breadth-first search...) from the states of X^i_{init} in the model Γ_i. The solution is the set of paths such that the sequence of observable events emitted by one path corresponds exactly to the sequence $\mathcal{O}_{\Gamma_i}$. In our tool, the algorithm of the local diagnosis computation is based on a diagnoser approach which is not detailed in this paper (see [8,13]). In brief, a local diagnoser is a data structure which is efficient to compute local diagnoses: it results from a compilation of the model of a component and directly maps the local observable events with the local trajectories which explain them.

3.2. *Merging local diagnoses to get the global diagnosis*

Once the local diagnoses are computed, the second phase is to merge the local results in order to obtain the global diagnosis. The basic operation is thus the merging operation which proceeds components two by two and is applied until the global diagnosis is obtained.

3.2.1. *The merging operation*

The merging operation takes as input two subsystems $\mathcal{I}$ and $\mathcal{J}$ and their diagnoses $\Delta_\mathcal{I}$ and $\Delta_\mathcal{J}$; it builds the diagnosis of the subsystem $\mathcal{I} \cup \mathcal{J}$ by composing the inputs, checking whether the interactions can be satisfied and eliminating all the invalid transitions. Initially, the subsystems are the components themselves. Step by step, the diagnosis for the global system is built by merging the diagnoses for bigger and bigger subsystems.

The merging operation consists thus in composing the finite-state machines representing diagnoses in order to obtain another finite-state machine in which only trajectories with valid interactions are represented. This merging operation is a classical synchronized product of two machines, [15,13] and is defined below.

Definition 4 (Merging operation) *Given* $\Delta_\mathcal{I} = (Q_\mathcal{I}, 2^{\Sigma^\mathcal{I}_{in}}, 2^{\Sigma^\mathcal{I}_{out}}, I_\mathcal{I}, F_\mathcal{I}, E_\mathcal{I})$ *the diagnosis of a subsystem* $\mathcal{I}$ *and* $\Delta_\mathcal{J} = (Q_\mathcal{J}, 2^{\Sigma^\mathcal{J}_{in}}, 2^{\Sigma^\mathcal{J}_{out}}, I_\mathcal{J}, F_\mathcal{J}, E_\mathcal{J})$ *the diagnosis of a subsystem* $\mathcal{J}$*, the merging of* $\Delta_\mathcal{I}$ *and* $\Delta_\mathcal{J}$ *is the finite-state machine :*

$$\Delta_\mathcal{I} \odot \Delta_\mathcal{J} = (Q_\mathcal{I} \times Q_\mathcal{J}, 2^{\Sigma^\mathcal{I}_{in} \cup \Sigma^\mathcal{J}_{in}}, 2^{\Sigma^\mathcal{I}_{out} \cup \Sigma^\mathcal{J}_{out}}, I_\mathcal{I} \times I_\mathcal{J}, F_\mathcal{I} \times F_\mathcal{J}, E_{\mathcal{I}\mathcal{J}})$$

where $E_{\mathcal{I}\mathcal{J}} \subseteq (Q_\mathcal{I} \times Q_\mathcal{J}) \times 2^{\Sigma^\mathcal{I}_{in} \cup \Sigma^\mathcal{J}_{in}} \times 2^{\Sigma^\mathcal{I}_{out} \cup \Sigma^\mathcal{J}_{out}} \times (Q_\mathcal{I} \times Q_\mathcal{J})$ *is the set of transitions such that :*

(*i*) $((x_1, x_2), I_1, O_1, (x'_1, x_2)) \in E_{\mathcal{I}\mathcal{J}}$ *if:*
$(x_1, I_1, O_1, x'_1) \in E_\mathcal{I}$ *and* $I_1 \cap \Sigma^\mathcal{J}_{out} = \emptyset \wedge O_1 \cap \Sigma^\mathcal{J}_{in} = \emptyset$;

(*ii*) $((x_1, x_2), I_2, O_2, (x_1, x'_2)) \in E_{\mathcal{I}\mathcal{J}}$ *if:*
$(x_2, I_2, O_2, x'_2) \in E_\mathcal{J}$ *and* $I_2 \cap \Sigma^\mathcal{I}_{out} = \emptyset \wedge O_2 \cap \Sigma^\mathcal{I}_{in} = \emptyset$;

(*iii*) $((x_1, x_2), I_1 \cup I_2, O_1 \cup O_2, (x'_1, x'_2)) \in E_{\mathcal{I}\mathcal{J}}$ *if:*
$(x_1, I_1 \cup I_\mathcal{I}, O_1 \cup O_\mathcal{I}, x'_1) \in E_\mathcal{I} \wedge (x_2, I_2 \cup I_\mathcal{J}, O_2 \cup O_\mathcal{J}, x'_2) \in E_\mathcal{J}$ *and*

$$O_1 \cap \Sigma_{in}^{\mathcal{J}} = \emptyset \wedge O_2 \cap \Sigma_{in}^{\mathcal{I}} = \emptyset \wedge I_1 \cap \Sigma_{out}^{\mathcal{J}} = \emptyset \wedge I_2 \cap \Sigma_{out}^{\mathcal{I}} = \emptyset \text{ and } \|(I_1 \cup I_2) \cap \Sigma_{fail}\| \leq 1 \wedge O_{\mathcal{I}} = I_{\mathcal{J}} \wedge O_{\mathcal{J}} = I_{\mathcal{I}}.$$

Transitions characterized by (i) and (ii) result from transitions of $\Delta_{\mathcal{I}}$ or $\Delta_{\mathcal{J}}$ which do not specify exchanged events between $\Delta_{\mathcal{I}}$ and $\Delta_{\mathcal{J}}$. Transitions characterized by (iii) are synchronized. They result from the synchronisation of one transition of $\Delta_{\mathcal{I}}$ and one of $\Delta_{\mathcal{J}}$ which exchange events $(I_{\mathcal{I}} = O_{\mathcal{J}} \wedge I_{\mathcal{J}} = O_{\mathcal{I}})$. The condition $(\|(I_1 \cup I_2) \cap \Sigma_{fail}\| \leq 1)$ expresses the fact that only one exogenous event of the system can occur at a given time as it is assumed in the model.

The result of merging the two diagnoses $\Delta_{\Gamma_i}(X_{init}^{\Gamma_i}, \mathcal{O}_i)$ and $\Delta_{\Gamma_j}(X_{init}^{\Gamma_j}, \mathcal{O}_j)$ is a finite-state machine which describes all the trajectories of Γ_i and Γ_j and such that the interactions between Γ_i and Γ_j are valid. Such a machine $(Q_{\Delta_i} \times Q_{\Delta_j}, 2^{\Sigma_{in}^i \cup \Sigma_{in}^j}, 2^{\Sigma_{out}^i \cup \Sigma_{out}^j}, I_{\Delta_i} \times I_{\Delta_j}, F_{\Delta_i} \times F_{\Delta_j}, E)$ describes exactly the diagnosis of Γ_i and Γ_j. This diagnosis is noted by $\Delta_{\Gamma_i\Gamma_j}(X_{init}^{\Gamma_i,\Gamma_j}, \{\mathcal{O}_i, \mathcal{O}_j\})$, shortly $\Delta_{\Gamma_i\Gamma_j}$. In the figure 4, the composition $\Delta_{\Gamma_l} \odot \Delta_{\Gamma_m}$ represents the set of all the trajectories of Γ_l and Γ_m explaining the observations from Γ_l and Γ_m (under the hypothesis that the interactions with other components are satisfied).

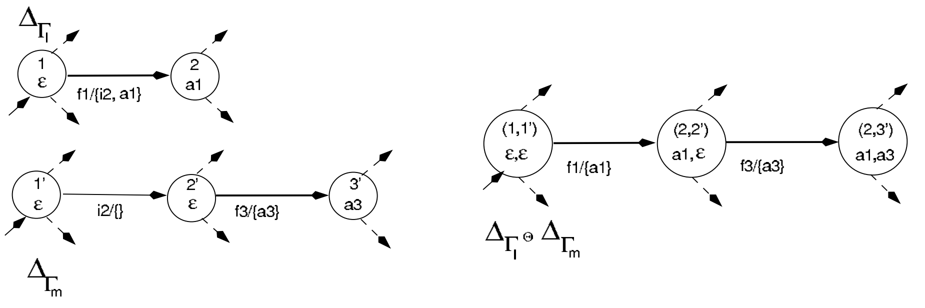

Fig. 4. Principle of the composition of two diagnoses: synchronization on the internal events. Here, Δ_{Γ_l} claims that $i2$ is exchanged between Γ_l and Γ_m. Δ_{Γ_m} confirms this interaction. $\Delta_{\Gamma_l} \odot \Delta_{\Gamma_m}$ is the diagnosis of the system composed by $\{\Gamma_l, \Gamma_m\}$.

3.2.2. *The global diagnosis*

By applying the $\odot$ composition on the local diagnoses, we finally obtain the diagnosis of the global system Γ. It is called the *global diagnosis.*

Definition 5 (Global diagnosis) *Given* $\Gamma = \{\Gamma_1, \dots, \Gamma_n\}$ *the model of the system,* X_{init}^{Γ} *a set of initial states of the system, and* $\mathcal{O} = \{\mathcal{O}_{\Gamma_1}, \dots, \mathcal{O}_{\Gamma_n}\}$ *a set of observation, the global diagnosis is defined as :*

$$\Delta_{\Gamma}(X_{init}^{\Gamma}, \mathcal{O}) = \bigodot_{i=1}^{n} \Delta_{\Gamma_i}(X_{init}^{\Gamma_i}, \mathcal{O}_{\Gamma_i})$$

where $X_{init}^{\Gamma_i}$ *is the set of initial states of* Γ_i *extracted from* X_{init}^{Γ}.

The global diagnosis is a communicating finite-state machine $\Delta_\Gamma(X^\Gamma_{init}, \mathcal{O})$, shortly Δ_Γ. The states Q_Δ are n-plets of the type $((s_{\Gamma_1}, \mathcal{E}_1), \ldots, (s_{\Gamma_n}, \mathcal{E}_n))$ where $s_{\Gamma_i} \in Q_i$ and where $\mathcal{E}_i$ is the prefix of the sequence $\mathcal{O}_{\Gamma_i}$ explained in this state. In other words, a state of the global diagnosis is defined by a state of the system $(s_{\Gamma_1}, \ldots, s_{\Gamma_n})$ associated with the set of observed sequences explained in this state. The initial states of Δ_Γ are those corresponding to X^Γ_{init} and the final states are such that $\forall i \in \{1, \ldots, n\}, \mathcal{E}_i = \mathcal{O}_i$. So the final states represent the current states of the system explaining the whole sequence of observations $\mathcal{O}$. The transitions are labeled with one failure exogenous event (from Σ_{fail}) as input : this is due to the fact that $\odot$ eliminates internal events (from Σ_{int}) and only accepts one failure event per transition. As the output events are concerned, the transitions are labeled only with a set of observed events ($\Sigma_{out} = \Sigma_{obs}$) for the same reason.

The data structure representing the global diagnosis contains all the needed information for the supervisor.

(i) Each transition path from an initial state to a final state is an explanation of the alarms by a sequence of failure occurrences in the system which permits a good interpretation of what has happened in the system.
(ii) The set of final states of the global diagnosis gives information about the state of the system after the observed alarms which is an interesting information from a monitoring point of view.

The interesting point in the diagnosis definition given above is that the $\odot$ operation is commutative and associative. Thus, the global diagnosis can be computed in several ways according to the order in which the diagnoses are progressively merged. The efficiency of the global computation depends directly on this key point which is detailed in the next section.

3.3. *Strategy for the merging operation*

In this section, we consider the problem of efficiently computing the global diagnosis by selecting the best order in which the merging operations are performed.

3.3.1. *Basic idea*

The strategy we proposed is based on the interactions between components claimed by their diagnoses. [13] Each diagnosis Δ_{Γ_i} contains trajectories which claim that the diagnosed component Γ_i has interacted with other components by sending or receiving internal events. Suppose that Δ_{Γ_i} claims that Γ_i emits an event $e_{i \to j}$ towards Γ_j. If Δ_{Γ_j} does not contain any trajectories which claim the reception of $e_{i \to j}$ then trajectories from Δ_{Γ_i} claiming the emission of $e_{i \to j}$ are *inconsistent* - they cannot participate to a global trajectory of the system. The merging of diagnoses consists in detecting such inconsistencies and eliminating the corresponding trajectories.

We focus in the following on the merging strategy and propose to base it on the interactions existing between the diagnoses.

3.3.2. *Definitions*

Each local diagnosis Δ_{Γ_i} contains trajectories which claim that the diagnosed component Γ_i has interacted with other components by sending or receiving events belonging to Σ_{int}. We note by $\mathcal{I}_{\Delta_{\Gamma_i}}(\Gamma_j)$ the set of events of Γ_i supposed to have been sent to or received from the component Γ_j according to the local diagnosis Δ_{Γ_i}.

Definition 6 (Inconsistent trajectories) *A* trajectory *in a diagnosis* Δ_{Γ_i} *is* inconsistent *iff there exists a transition in this trajectory which assumes the emission or the reception of an event* $e \in \mathcal{I}_{\Delta_{\Gamma_i}}(\Gamma_j)$ *such that* $e \notin \mathcal{I}_{\Delta_{\Gamma_j}}(\Gamma_i)$.

A diagnosis Δ_{Γ_i} may claim that an event e belongs to $\mathcal{I}_{\Delta_{\Gamma_i}}(\Gamma_j)$ whereas Δ_{Γ_j} may claim that e is not in $\mathcal{I}_{\Delta_{\Gamma_j}}(\Gamma_i)$. Trajectories of Δ_{Γ_i} that claim the occurrence of e are said to be inconsistent: they will not participate to the global diagnosis and can be immediately discarded from the diagnosis.

In the following, we call *purged diagnosis* of Γ_i, the set of trajectories of Δ_{Γ_i} from which inconsistent trajectories have been eliminated and note it by Δ'_{Γ_i}. [2]

Definition 7 (Matchable components) Γ_i *and* Γ_j *are* matchable components *iff*

$$\mathcal{I}_{\Delta_{\Gamma_i}}(\Gamma_j) \cap \mathcal{I}_{\Delta_{\Gamma_j}}(\Gamma_i) \neq \emptyset.$$

From this definition, it can be deduced that Γ_i and Γ_j are matchable iff their purged diagnoses are such that $\mathcal{I}_{\Delta'_{\Gamma_i}}(\Gamma_j) = \mathcal{I}_{\Delta'_{\Gamma_j}}(\Gamma_i) = \mathcal{I}_{\Delta_{\Gamma_i}}(\Gamma_j) \cap \mathcal{I}_{\Delta_{\Gamma_j}}(\Gamma_i) \neq \emptyset$. In the following, we will say that Γ_i and Γ_j are *k-matchable* components iff they are matchable and $\mid \mathcal{I}_{\Delta'_{\Gamma_i}}(\Gamma_j) \mid = \mid \mathcal{I}_{\Delta'_{\Gamma_j}}(\Gamma_i) \mid = \mid \mathcal{I}_{\Delta_{\Gamma_i}}(\Gamma_j) \cap \mathcal{I}_{\Delta_{\Gamma_j}}(\Gamma_i) \mid = k$. By extension of the definition, we say that two distinct sets of components γ_1, γ_2 are k-matchable iff $\sum_{(\Gamma_i,\Gamma_j),\Gamma_i \in \gamma_1,\Gamma_j \in \gamma_2} \mid \mathcal{I}_{\Delta'_{\Gamma_i}}(\Gamma_j) \mid = k, k > 0$.

3.3.3. *Principles and algorithm*

To improve the efficiency of the merging phase, we apply the two following principles (see the algorithm in figure 5).

1. *Detecting and eliminating inconsistent trajectories.* Inconsistent trajectories uselessly increase the cost of the composition operation. The first principle consists in detecting and eliminating them before any composition operation is performed.

2. *Giving priority to the most matchable components.* The merging of two diagnoses allows to eliminate some trajectories by checking the interactions which are claimed by them. Thus, merging two diagnoses which do not claim any interaction between their respective components is not really interesting: the second principle consists in avoiding this unuseful computation and in giving priority to the most matchable components.

   The merging strategy builds partition of diagnoses such that:

   - a partition element has two diagnoses at the most;

[2] The model is supposed to be complete so that such a diagnosis always exists.

```
**Input** :$\{\Delta_{\Gamma_i}, i \in \{1,\dots,n\}\}$
**Input** :$\{\mathcal{I}_{\Delta_{\Gamma_i}}(\Gamma_j), i,j \in \{1,\dots,n\}, j \neq i\}$
**1 − Incompatible trajectory elimination**
for $i \leftarrow 1$ to $n$ do
    $\Delta_{tmp} \leftarrow \Delta_{\Gamma_i}$
    for $j \leftarrow 1$ to $n, j \neq i$ do
        $badEvnt \leftarrow \mathcal{I}_{\Delta_{\Gamma_i}}(\Gamma_j) - \mathcal{I}_{\Delta_{\Gamma_j}}(\Gamma_i)$
                                {badEvnt: events responsible of incompatible trajectories in $\Delta_{\Gamma_i}$
                                                                                    according to $\Delta_{\Gamma_j}$}
        $\mathcal{I}_{\Delta'_{\Gamma_i}}(\Gamma_j) \leftarrow \mathcal{I}_{\Delta_{\Gamma_i}}(\Gamma_j) - badEvnt$
        $\Delta_{tmp} \leftarrow ElimImposTraj(\Delta_{tmp}, badEvnt)$
                                  {$\Delta_{tmp}$ has no impossible trajectories more according to $\Delta_{\Gamma_j}$
    end
    $\Delta'_{\Gamma_i} \leftarrow \Delta_{tmp}$
end
**2 − Looking for matchable components**
for $i \leftarrow 1$ to $n$ do
    $kInter(\Gamma_i) \leftarrow \{(\Gamma_j, \mathcal{I}_{\Delta'_{\Gamma_i}}(\Gamma_j)), \mathcal{I}_{\Delta'_{\Gamma_i}}(\Gamma_j) \neq \emptyset\}$
end
**3 − Composition strategy in parallel**
$\mathcal{D} \leftarrow \{\Delta'_{\Gamma_i}, i \in \{1,\dots,n\}\}$
$\mathcal{M} \leftarrow \{kInter(\Gamma_i), i \in \{1,\dots,n\}\}$
while $\mathcal{M} \neq \emptyset$ do
    $\pi_{\mathcal{D}} \leftarrow ChoosePartitionWithInter(\mathcal{D}, \mathcal{M})$
    $\mathcal{D} \leftarrow ComposeInParallel(\pi_{\mathcal{D}})$
    $\mathcal{M} \leftarrow UpdateInter(\mathcal{M}, \mathcal{D})$
end
**{The set $\mathcal{D}$ represents the global diagnosis}**
```

Fig. 5. Global diagnosis computation algorithm.

- selected diagnoses are such that the set of exchanged events claimed by those diagnoses is as big as possible.

Once a partition of diagnoses is chosen, the diagnoses of each element of the partition are composed in a parallel way. A new set of diagnoses is obtained where one diagnosis is associated to each element of the partition. The set of possible exchanged events is updated according to the new diagnoses set. Then, a new stage proceeds by building the best new partition of diagnoses and composing it. The last stage produces a set of diagnoses which claim that there is no possible exchanged events between them. Each diagnosis of the resulting set corresponds to a distinct group of components in the system and implicitly represents the global diagnosis. Because those diagnoses suppose that there are no exchanged events between components of two different groups, the composition is unnecessary : the diagnoses obtained in this last stage are said to be *independant*. The trajectories of every diagnosis of the resulted set participate to a global trajectory.

4. Application to telecommunication networks

The purpose of our diagnosis system is to deal with telecommunication networks. In this section, we present a subpart of a real packet-switching telecommunication network (see figure 6). It is composed of eight switches (*Sw*) which are managed by two different technical centers (*Tc*). Each switch is associated with four computer stations. There are two kind of stations: operation stations and managing stations. The operating stations are used to run basic applications of the switch: each switch is composed of two operating stations (primary station *Os1*, secondary station *Os2*). When no failure occurs on the stations, only the primary station works, the secondary station being in passive mode. If a failure occurs on this station, a mechanism swaps the stations (the secondary station begins to work) in order to always have an operationnal switch. The managing stations are used to control the links between the different switches (routing). For the same reason as the operating stations, each switch is composed of two managing stations (primary and secondary *Ms1,Ms2*). The described network is then composed of 42 components.

In this architecture, alarms from stations are emitted via the associated switch which means that if the switch is down, the alarms emitted by the stations are lost: this phenomenon is called *masking phenomenon*.

A technical center may emit two kinds of alarms: *cvhs* and *cves*. *cvhs* is emitted when the technical center goes down, due to a rebooting or a connection cut. *cves* is emitted when the technical center begins to work well again. A switch may also emit two kinds of alarms : *n003* and *n004*. *n003* is due to a blocking of the switch or a booting ordered by its technical center and *n004* is emitted when the switch begins to work well again. As the stations are concerned, they emit several alarms of type *p089*. These alarms are emitted when the stations are rebooting, stopping or when there is a swap between two stations of the same type.

In figure 7, the model of a primary managing station (*Ms1*) is presented. A primary

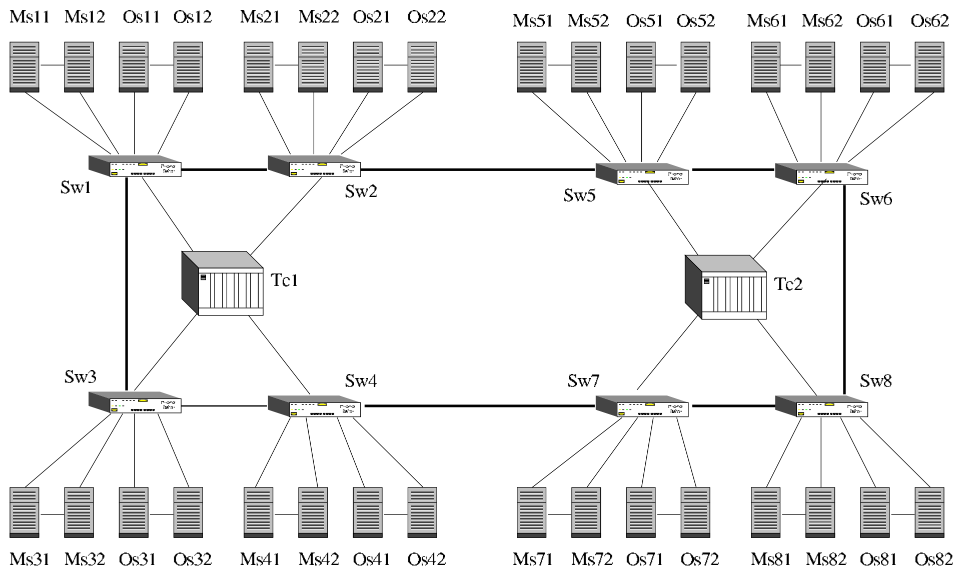

Fig. 6. Topology of the studied network.

station can be active, rebooting, stopping, moreover it can be independently masked or not, so six states are distinguished in its behavior. The failure events that can occur on the station are *blkMs1* (the station begins to stop), *rebootMs1* (the station begins to reboot). These failure events are associated with back events, respectively *backMs1* and *enrbtMs1*, which model the fact that failures are not permanent. *Ms1* can interact with the associated switch (*Sw*) and the secondary station (*Ms2*) (see figure 8). In the case of a *blkMs1* event, there is a swap of the primary and secondary stations which is modeled by the *swpMs2* event emitted by the model of *Ms1* and received by the model of *Ms2*. When *Ms1* begins to work well again (*backMs1*), *Ms2* becomes passive which corresponds to the event *swpMs1*. The masking phenomenon is modeled with the help of the *mskMs1* and *unMskMs1* events which are taken into account by the station when the associated switch masks or unmasks the station.

The difficulty of diagnosing this system is that one alarm is not sufficient to identify a failure (for example, *n003* corresponds to a rebooting or a blocking). Discrimination requires to look at the occurrence of other alarms.

5. D-Dyp: a telecommunication diagnosis tool

This tool implements all the tasks needed for the on-line diagnosis computation of systems like telecommunication networks given a model description. The diagnosis task is composed of two steps:

(i) an *off-line* step which consists in the acquisition of the model and the compilation of

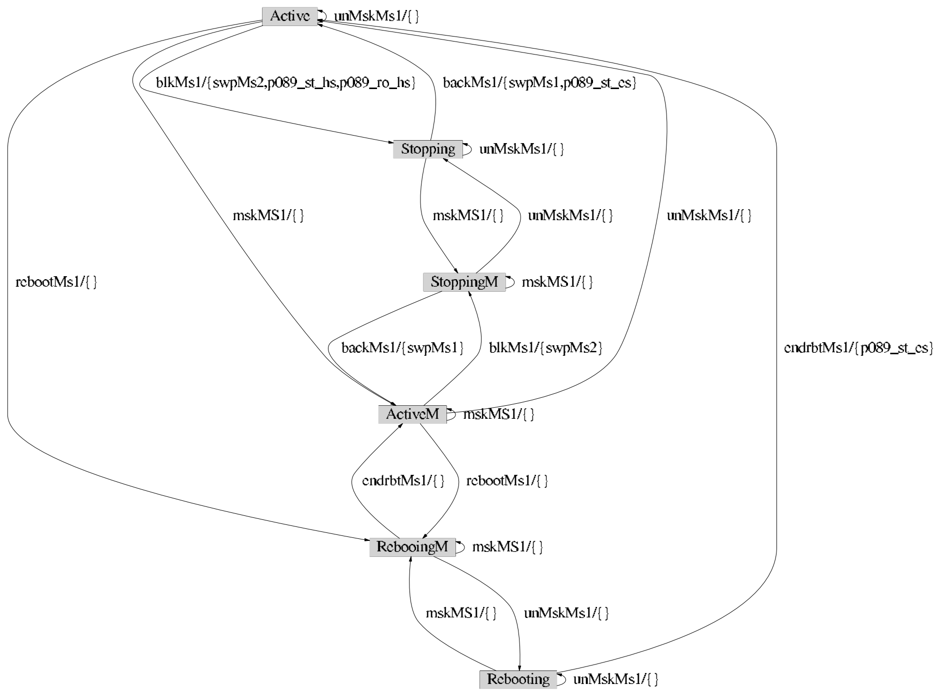

Fig. 7. Model of a managing station (*Ms1*).

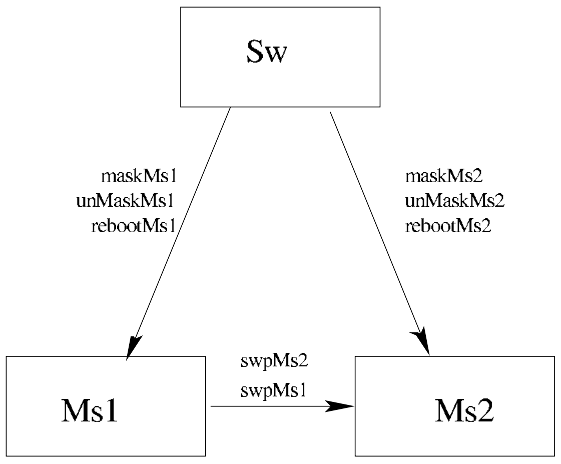

Fig. 8. Interactions between *Ms1*, *Ms2* and *Sw*.

the local diagnosers from a model description file;

(ii) an *on-line* step which consists in the acquisition of the alarms and the computation of the global diagnosis.

5.1. *Modeling*

The model is described with a language of description derived from the ESTELLE language. [18] This language permits to express the behavior of the different components of the system. For exemple, the skeleton of the primary station behavior is depicted below.

```
MODULE Ms1;
/* primary station model */
  IP
    INPUT Swi : (reboot,masque,demasque);
    OUTPUT obs : (p089_st_hs,p089_ro_hs,p089_st_es);
    INPUT exo : (blkMs1,backMs1,endrbtMs1);
    OUTPUT secondary : (swpMs1,swpMs2);
  END;

  BEHAVIOR body_Ms1 FOR Ms1;
    STATE Active,Stopping,Rebooting,ActiveM,StoppingM,RebootingM;
    ...
    TRANS
       FROM Active TO Stopping
       WHEN exo.blkMs1
       OUTPUT secondary.swpMs2
       OUTPUT obs.p089_st_hs
       OUTPUT obs.p089_ro_hs
    ;
    ...
  END;
END;
```

The structural model describing the interactions between several components are expressed with the help of connections between modules. For example, the connection between an *Ms1* and an *Ms2* is given below:

```
CONNECT Ms1.secondary TO Ms2.primary;
```

This model description file can be obtained by the use of a graphical model editor which has been implemented for this task. Another way to obtain this description is to automatically acquire it from a model in another equivalent formalism like SDL or UML.

5.2. *Diagnosis computation*

The diagnosis platform we used for on-line diagnosing is a distributed application. Given the local diagnosers computed in the off-line step, we instanciate the diagnosis platform as a set of CORBA objects using the client/server paradigm (see figure 9).

This platform is composed of four kinds of objects :

- **Local diagnoser i.** This CORBA object implements the local diagnoser. The purpose of this CORBA object is to look at the network, to capture the alarms emitted

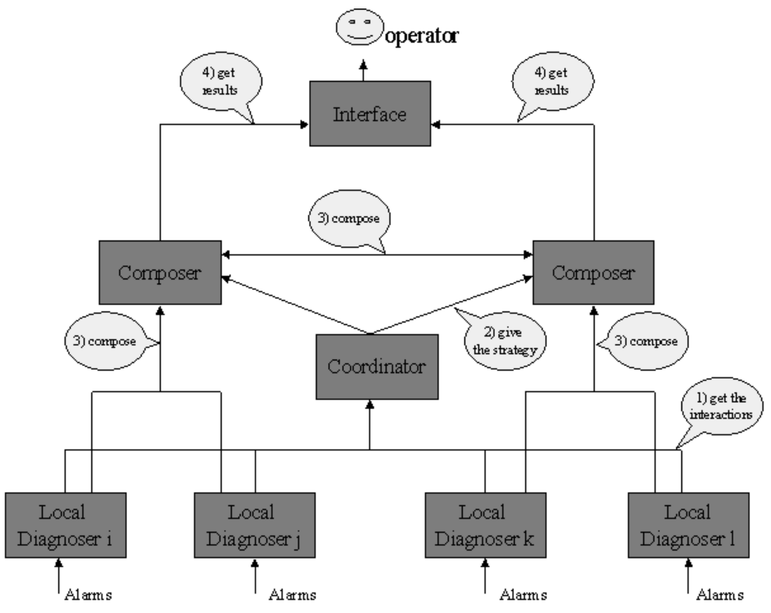

Fig. 9. Distributed diagnosis platform.

by the associated component Γ_i and and to update its local diagnosis.

- **Coordinator.** The coordinator task is to compute the strategy of merging by taking into account the current interactions claimed by each local diagnoser. Then, it calls the composer objects which apply the strategy.
- **Composer.** The task of a composer object is to apply the $\odot$ composition operation. According to the coordinator, it gets the diagnoses from the local diagnosers or from other composers and applies the composition.
- **Interface.** This object is the interface of the tool. It permits to visualize the current diagnoses of the system and to display them to the operator.

There are several advantages of using a distributed architecture. Firstly, we can deploy each local diagnoser on a machine well-adapted for the observation of a given set of components (machine close to these components...). Secondly, the use of several composers permits to compose the diagnoses in a real parallel way (one composer per processor) and increases the efficiency of the merging operation.

6. Results

In this experiment, we have computed the global diagnosis in a temporal window containing 56 alarms (see table 1) emitted by 20 components of the network presented in section 4.

Table 1. The set of alarm sequences observed during one temporal window.

Os11	Os12	Ms21	Ms22
p089_2_st_hs	p089_2_ro_es	p089_ro_hs	p089_sec_st_hs
p089_2_st_es	p089_1_ro_es	p089_st_hs	p089_nom_st_es
p089_2_ro_es	p089_2_st_hs	p089_st_es	p089_nom_ro_rope
p089_2_ro_hs			p089_nom_st_hs
p089_1_ro_es			p089_nom_ro_nop
			p089_sec_st_es
			p089_sec_st_hs
			p089_sec_ro_hs

Sw4	Ms41	Ms42	Os41
n003	p089_st_es	p089_sec_st_hs	p089_1_ro_es
n004		p089_sec_ro_hs	

Os42	Sw5	Os51	Os52
p089_2_st_es	n003	p089_2_st_hs	p089_2_ro_hs
	n004	p089_2_st_es	p089_1_ro_es
		p089_2_ro_es	p089_2_st_hs
		p089_2_ro_hs	
		p089_1_ro_es	

Sw6	Ms61	Ms62	Sw8
n003	p089_st_hs	p089_sec_st_hs	n003
n004	p089_ro_hs	p089_nom_st_es	n004
	p089_st_es	p089_nom_ro_rope	
		p089_nom_st_hs	
		p089_nom_ro_nop	
		p089_sec_st_es	
		p089_sec_st_hs	
		p089_sec_ro_hs	

Ms81	Ms82	Os81	Os82
p089_st_es	p089_sec_st_hs	p089_1_ro_es	p089_2_st_es
	p089_sec_ro_hs		

6.1. *Analyses of the result*

For this experiment, the diagnosis platform uses three composer objects in order to parallelize some computations. The number of composers to instantiate depends of course of the complexity of the diagnosed systems and on the number of processors we can use for the diagnosis computation.

As the local diagnoses computation is concerned, dealing with this set of observations is very efficient. In fact, the time of computation is inferior to 100ms for each local diagnosis. This efficiency is due to the diagnoser approach implemented in the tool.

According to the received alarms, no interaction is possible between the technical centers *Tc1, Tc2* and their switches *Swi*. However, the local diagnoses of the switches claim such an interaction (they suppose that a masking phenomenon has occurred). After the elimination of these incompatible trajectories, the local purged diagnoses of the switches become independant of the diagnoses of the technical centers and no merging is needed (*Tc1* and *Tc2* are independant). For the same reason $\Delta_{Ms71}, \Delta_{Ms72}, \Delta_{Os71}, \Delta_{Os72}$ are independant from Δ_{Sw7} and are not merged. The result of the diagnosis computation is then a set of eleven independant diagnoses.

$$\{\Delta_{Tc1}, \Delta_{Sw1,Ms11,Ms12,Os11,Os12}, \Delta_{Sw2,Ms21,Ms22}, \Delta_{Os21}, \Delta_{Sw3,Ms31,Ms32,Os31,Os32}, \Delta_{Sw4,Ms41,Ms42,Os41,Os42}, \Delta_{Tc2}, \Delta_{Sw5,Ms51,Ms52,Os51,Os52}, \Delta_{Sw6,Ms61,Ms62,Os61,Os62}, \Delta_{Sw7}, \Delta_{Ms71}, \Delta_{Ms72}, \Delta_{Os71}, \Delta_{Os72}, \Delta_{Sw8,Ms81,Ms82,Os81,Os82}\}.$$

The result has been computed in 8 seconds (real time).

6.2. *Comparisons with other strategies*

In order to show the efficiency of our strategy, we have compared it to other strategies of computations. This comparison is established by the computation of the global diagnosis relative to a subpart of the presented network using only one composer object (no parallelism). The subpart of the network is composed of *Tc2, Sw8, Ms81, Ms82, Os81, Os82.* Only the alarms emitted by this set of components and presented in the table 1 are considered in this experiment. Times of computations (CPU times) are presented in the figure 10. For each stage of merging (here 5 stages), the figure presents the cumulative time since the first stage.

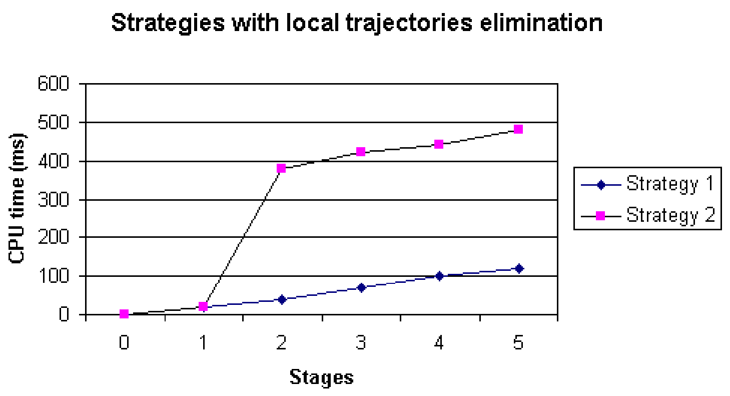

Fig. 10. Comparisons of the performance of two different strategies of merging.

Strategy 1 is the strategy used by the diagnostic tool when one composer is available. For a better comparison with other strategies, strategy 1 also computes the merging of independant diagnoses (at stage 5) in order to explicitly obtain the global diagnosis. Strategy 2 eliminates inconsistent trajectories but do not apply any merging strategy based on interacting diagnoses. We have also experimented strategies which do not apply any inconsistent trajectory elimination, in this case, the computation requires several hours !

7. Conclusion

This paper describes a model-based diagnostic tool especially adapted to large and evolutive discrete-event systems such as telecommunication networks. Dealing with global models is clearly impossible with such systems. The idea is to compute local diagnoses from component models and to build the global diagnosis by merging these local results. The challenge is to preserve the efficiency of this global on-line computation. One key point is considering with this respect: the order in which the successive merging operations are performed. Our tool implements a strategy favoring the diagnoses corresponding to interacting components. This strategy can be compared to the reconstruction plan.[11] The main advantage of our proposal is that the interactions are looked for by an on-line analysis of diagnoses making this strategy dynamic and adaptable.

Several improvements are currently studied in order to use the diagnostic on larger systems. An incremental algorithm able to take into account observations on successive temporal windows has been designed. [14] It is currently experimented with satisfying results. An important point is the choice of the size of the temporal window in which the strategy of merging is applied. The optimal size of the temporal windows has still to be determined, knowing that it clearly depends on the application requirements.

Acknowledgements

This work is partially supported by RNRT (French National Research in Telecommunication) through the MAGDA project (Modeling and Learning for Distributed Management of Alarms). For more information, see http://magda.elibel.tm.fr/.

[1] D. Niebur. Expert systems for power system control in western europe. *Proceedings of the IEEE Symposium on Intelligent Control*, pages 112–119, 1990.

[2] C. Dousson, P. Gaborit, and M. Ghallab. Situation recognition: representation and algorithms. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 166–172, Chambéry, France, 1993.

[3] M.-O. Cordier and C. Dousson. Alarm driven monitoring based on chronicles. In *Proceedings of Safeprocess'2000*, pages 286–291, june 2000.

[4] M. Riese. Diagnosis of extended finite automata as a dynamic constraint satisfaction problem. In *Proceedings of the International Workshop on Principles of Diagnosis(DX'93)*, pages 60–73, Aberystwyth, UK, 1993.

[5] J. Lunze. Discrete-event modelling and diagnosis of quantized dynamical systems. In *Proceedings of the International Workshop on Principles of Diagnosis (DX'99)*, pages 147–154, Loch Awe, 1999.

[6] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete event system. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, 1995.

[7] L. Rozé and M-O. Cordier. Diagnosing discrete event sytems : An experiment in telecommunication networks. In *WODES98, Fourth Workshop on Discrete Event Systems*, Cagliari, Italy, 1998.

[8] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Active diagnosis of discrete-event systems. *IEEE Transactions on Automatic Control*, 43(7):908–929, 1998.

[9] L. Rozé and M.-O. Cordier. Diagnosing discrete-event systems : extending the "diagnoser approach" to deal with telecommunication networks. *Journal on Discrete-Event Dynamic Systems (JDEDS)*,12,1,January 2002.

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

[11] P. Baroni, G. Lamperti, P. Pogliano, and M. Zanella. Diagnosis of large active systems. *Artificial*

Intelligence, 110:135–183, 1999.

[12] L. Console, C. Picardi, and M. Ribaudo. Diagnosis and diagnosibility analysis using pepa. In *Proceedings of the European Conference on Artificial Intelligence (ECAI'2000)*, pages 131–135, Berlin, 2000.

[13] Y. Pencolé. Decentralized diagnoser approach: application to telecommunication networks. In *Proceedings of the International Workshop on Principles of Diagnosis (DX'00)*, pages 185–192, Morelia, Mexico, 2000.

[14] Y. Pencolé, M-O. Cordier, and L. Rozé. Incremental decentralized diagnosis approach for the supervision of a telecommunication network. In *Proceedings of the International Workshop on Principles of Diagnosis (DX'01)*, pages 151–158, Sansicario, Italy, 2001.

[15] A. Arnold. Transition systems and concurrent processes. In *Mathematical problems in Computation theory*, volume 21. Banach Center, 1987.

[16] M-O. Cordier and S. Thiébaux. Event-based diagnosis for evolutive systems. In *Proceedings of the International Workshop on Principles of Diagnosis (DX-94)*, pages 64–69, New Palz (NY), October 1994.

[17] C. Barral, S. McIlraith, and T. C. Son. Formulating diagnostic problem solving using an action language with narratives and sensing. In *Proceedings of KR'2000)*, pages 311–322, Cambridge, 2000.

[18] ISO. Information processing systems — Open systems interconnection — Estelle — a formal description technique based on an extended state transition model, 1997.