HAL
open science

Diagnosing discrete-event systems: extending the "diagnoser approach" to deal with telecommunication networks

Laurence Rozé, Marie-Odile Cordier

To cite this version:

Laurence Rozé, Marie-Odile Cordier. Diagnosing discrete-event systems: extending the "diagnoser approach" to deal with telecommunication networks. [Research Report] RR-4957, INRIA. 2003. inria-00071622

HAL Id: inria-00071622

https://inria.hal.science/inria-00071622v1

Submitted on 23 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Diagnosing discrete-event systems: extending the "diagnoser approach" to deal with telecommunication networks

Laurence Rozé, Marie-Odile Cordier

N° 4957

Septembre 2003

THÈME 3

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--4957--FR+ENG

Diagnosing discrete-event systems: extending the "diagnoser approach" to deal with telecommunication networks

Laurence Rozé, Marie-Odile Cordier

Thème 3 — Interaction homme-machine,
images, données, connaissances
Projet Dream

Rapport de recherche n° 4957 — Septembre 2003 — 46 pages

Abstract: Detection and isolation of failures in large and complex systems such as telecommunication networks are crucial and challenging tasks. The problem considered here is that of diagnosing the largest French packet switching network. The challenge is to be as efficient as the existing expert system while providing greater generality and flexibility with respect to technological and reconfiguration changes in the network. The network is made up of interconnected components each of which can send, receive and transmit messages via its ports. The problem we are faced with is to follow the evolution of the network on the basis of the stream of time-stamped alarms which arrive at the supervision center. We have decided to use model-based techniques which are recognized to be more adapted to evolutive systems than expertise-based approaches are.

This paper starts with a description of how we model the global behavior of this discrete-event system by using communicating finite state machines. It goes on to explain how this model is used for analyzing the stream of alarms and diagnosing the network. Our work is based on the diagnoser approach proposed by [20]. Starting from a model of the network adapted to simulate faults, this approach transforms it into a finite state automaton, called a diagnoser, in order to analyze the stream of alarms. The approach described in [20, 21] proved to be grounded on certain basic hypotheses which were too restrictive for our application. This paper extends [21]'s proposal to communicating finite state machines. The difficulties we had to cope with are outlined and the way we overcome them is presented. A major difficulty is the huge size of the global model of the system. To solve this problem we take advantage of the hierarchical structure of the network and rely on a generic model of the system for building a generic diagnoser.

Thanks to the reviewers for their constructive comments. Special thanks to Yannick Pencolé for his involvement in the implementation phase and his great help to revise a first version of this paper.

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex (France)
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

Key-words: model based approach, real-time monitoring, telecommunication networks, communicating finite state machine, generic model

Diagnostique de systèmes à événements discrets : extension de l'approche diagnostiqueur pour la supervision de réseaux de télécommunications

Résumé : La détection de pannes dans des systèmes complexes tel que les réseaux de télécommunication est un problème difficile et crucial. L'application étudiée dans cet article est la détection de pannes dans le plus grand réseau français à commutation de paquets. Le challenge est d'être aussi efficace que le système expert existant et d'offrir une plus grande adaptativité (prise en compte des évolutions technologiques et des reconfigurations du réseau). Ce dernier est constitué d'un ensemble de composants interconnectés entre eux. Chaque composant peut émettre, recevoir, ou transmettre des messages. Le problème est de suivre l'évolution du réseau à partir du flot d'alarmes datées arrivant au superviseur. Nous utilisons des techniques à base de modèles qui sont plus adaptées aux systèmes évolutifs que les systèmes experts.

Dans un premier temps nous décrivons comment nous modélisons le comportement global du système à l'aide d'automates communicants. Puis nous expliquons comment ce modèle est utilisé pour analyser le flot d'alarmes et effectuer le diagnostic. Nos travaux s'appuient sur l'approche diagnostiqueur proposée par [20]. Partant d'un modèle de réseau adapté à la simulation de panne, cette approche transforme le modèle en un automate fini appelé diagnostiqueur. L'approche proposée dans [20, 21] s'appuie sur des hypothèses trop restrictives pour notre application. Notre papier étend [21] aux automates communicants. Les difficultés rencontrées et les solutions utilisées sont présentées. La plus grande difficulté était liée à la taille du système à diagnosquer. Pour résoudre ce problème nous avons tiré profit de la structure hiérarchique du réseau et nous avons construit un modèle et un diagnostiqueur générique.

Mots-clés : techniques à base de modèles, supervision, réseaux de télécommunications, automates communicants, modèle générique

Introduction

Detection and isolation of failures in large and complex systems such as telecommunication networks are crucial and challenging tasks. The problem considered here is that of diagnosing the largest French packet switching network[1]. An expert system currently helps operators to analyze streams of time-stamped events (or alarms) and identify problems on the network. Every day around one hundred thousand alarms are received and analyzed by the supervision center. Our challenge is to be as efficient as the existing expert system while providing greater generality and flexibility with respect to technological and reconfiguration changes in the network.

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 [14] or scenario recognition systems [12]), and model-based techniques which reason abductively directly on the model of the system ([17]). 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.

The approach we have used for monitoring the network is based on the methodology proposed in [20, 21] known as "the diagnoser approach". It consists in transforming the discrete event model of the system to be diagnosed into a finite state automaton, called a diagnoser. The transitions of this automaton are the observable events, and the states of the automaton contain information on the state of the system and on the failure(s) that must have occurred in order to be in this state.

The diagnoser approach is attractive for two reasons. First, it is efficient, since diagnosing consists in updating the current state of the finite state machine according to observables. Second, it is adaptative; if the system changes (structural or functional changes), the model is consequently updated and the diagnoser derived automatically. However, when trying to apply this diagnoser approach as described in [21], it rapidly becomes clear that it was grounded on basic hypotheses which were too restrictive for our application; since it relied on a classical finite-state machine formalism, it was first necessary to extend it to cope with a communicating finite-state machine formalism which is particularly well suited for the modeling of telecommunication networks. Other specificities also had to be taken into account, including the existence of intermittent faults, the importance of temporal information and, finally, the untractable size of the global model of the system. This paper analyzes these difficulties and explains how we overcame them.

Section 1 describes the application. Section 2 shows how we model the network to be diagnosed by using two kinds of models at two levels of abstraction: a structural model represented by a labeled graph and a behavioral model represented by temporal communicating finite-state machines. Section 3 recalls the diagnoser approach as originally presented in [21]. Section 4 explains how it was extended to cope with the specificities of our application. Sec-

[1]This research was carried out in a joint project with the CNET France Telecom (Centre National d'Etudes des Télécommunications), in the CNET/CNRS 93 1B 142 513 project.

tion 5 focuses on the problematic size of the model and explains how we have solved it by taking advantage of the hierarchical structure of the network and using a "generic" model. The implementation and the results we obtain with the diagnoser approach are presented in section 6. Section 7 compares our work to related work and section 8 concludes with the current state of this work and future perspectives.

1 The application

1.1 The network

The network considered in this paper is the largest French packet switching network. This network is made up of about ten technical centers (TC) and about three hundred switches (SW), hierarchically structured (cf figure 1). SC is the supervision center. The switches route data through the network. They are made up of stations (hardware) on which roles (software) run. There are two kinds of roles and stations: UE and USG. The UE role is the operating system of the switch. If a breakdown occurs on the UE station (which is the station on which the UE role runs), the switch is no longer working. For safety, an SW includes two UE stations and roles; one is active and the other will become active if and only if the active UE station breaks down. The USG stations and roles are in charge of data transmission. A rescue station is associated to a group of USG stations. When a USG station breaks down, its role will run on its rescue station, provided it is not busy.

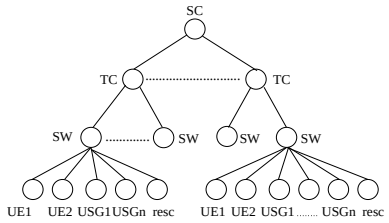


Figure 1: Hierarchical structure of the network

1.2 The supervision problem

Each time an event occurs in the network, messages are sent by components either to other components or to the supervision center itself. In this last case they are called alarms. Alarms are forwarded from one component to another up to the supervision center (SC). For example, an alarm emitted by an SW is transmitted by a TC before arriving at the SC. The supervision problem consists in analyzing the stream of alarms arriving at the supervision center (about 100,000 alarms can be received in a day) in order to follow the evolution of the network and to help the operator who is in charge of the supervision. In

particular, the occurring faulty situations have to be identified. The failures we take into account are listed below:

- TC_cut, which corresponds to the breaking of the link between a technical center and the supervision center.
- TC_break, which corresponds to the breakdown of a technical center. The consequence of this breakdown is the rebooting of all the switches depending on it.
- SW_cut, which corresponds to the breaking of the link between a switch and the technical center it depends on.
- SW_break, which corresponds to the breakdown of a switch. The consequence of this breakdown is the rebooting of all the stations depending on it.
- USG_shift, which corresponds to the breakdown of a USG station. As a consequence, the role running on it shifts to the rescue station (provided it is not already busy).
- $USG_reconfiguration$, when a USG role returns from the rescue station to its nominal station.
- USG_rescue_break, which corresponds to the breakdown of a rescue station.
- USG_break, which corresponds to the breakdown of a USG station when the rescue station is not free.
- UE_shift, which corresponds to the breakdown of the active UE station. As a consequence, the UE role shifts to the other UE station.
- UE_break, where the active UE station breaks down whereas the other UE station has already broken down;
- UE_rescue_break, which corresponds to the breakdown of the UE station which is not currently in charge of the switch.

These failures are not permanent failures; they disappear after a while either due to automatic recovery procedures or due to repair actions. In the first case, the time taken by the recovery procedure is approximatively known. In the second case, the "comeback" to a normal state corresponds to an external event. For example:

- TC_SC_back denotes an external event corresponding to the comeback to a normal state of a technical center after the occurrence of a TC_cut or of a TC_break. In the following, it will be also designed by TC_back.
- SW_TC_back denotes an external event corresponding to the comeback of a switch after the occurrence of a SW_cut or of a SW_break.

- TC_SW_back signals the end of the rebooting of a switch which is an automatic recovery procedure triggered by a TC_break.

Note that the supervision center is limited to the supervising of the components of the network and it is not in charge of supervising the data transmission itself.

1.3 Difficulties

The supervision center receives about 100,000 alarms a day. Analyzing this sequence of alarms is a difficult task, made harder by two phenomena: the masking phenomenon and the necessity of taking into account temporal information.

When a component sends an alarm, this alarm has to be transmitted by components before arriving at the SC. If one of the components in the transmission path is not in a transmitting state, the alarm will not be received by the supervision center. For example, if an SW sends an alarm and if the TC just above is in a non-transmitting state, the alarm will not be received by the supervision center. One of the main difficulties is thus the incompleteness of the stream of events arriving at the SC.

The second difficulty is the necessity of taking into account temporal information. It can be the case that two faults cannot be discriminated without using temporal constraints. It is especially so in the case of masking. According to the time taken by a component to return to a normal state, an alarm resulting from a fault will or will not reach the supervisor. Moreover, this temporal information is often imprecise; for example, when a TC_break occurs, a TC_SW_back is an automatic recovery, corresponding to the SW reboot, whose time is approximatively known and fluctuates.

Let us take an example including both masking and temporal constraints. When a TC_break occurs, both the technical center TC and the switches SW depending on it are going to break down (each SW reboots). When a technical center breaks down, it sends a *cvhs* alarm to the supervisor; when it returns to a normal state, it sends a *cves* alarm. When an SW breaks down, it sends an $n003$ alarm and when it returns to a normal state, it sends an $n004$ alarm to the supervision center. These alarms have to be transmitted through the technical center before arriving at the supervision center (see figure 1) and this technical center has to be in a normal state to be able to transmit them. In the case of a TC_break, the $n003$ alarms have no chance of reaching the supervisor, they are going to be masked. Concerning the $n004$ alarms, it depends on the respective dates of the return to a normal state of the technical center and of the switch. If the switch comes back to a normal state before the technical center does, the $n004$ alarm will be masked; in the other case, it will reach the supervisor.

Let us suppose two switches SW1 and SW2 under the technical center TC. According to the respective times of the SW1, SW2 and TC "comebacks", the alarm sequences for a TC_break can be one of the following ones:
– $cvhs(TC)$, $cves(TC)$, $n004(SW1)$, $n004(SW2)$
– $cvhs(TC)$, $cves(TC)$, $n004(SW2)$, $n004(SW1)$
– $cvhs(TC)$, $cves(TC)$, $n004(SW1)$

– $cvhs(TC)$, $cves(TC)$, $n004(SW2)$
– $cvhs(TC)$, $cves(TC)$
In the third sequence, SW2 completed its rebooting before the technical center comebacks in a normal state: its $n004$ alarm is masked; it is not the case for $SW1$ and its $n004$ alarm is received by the supervision center. In the last sequence, the two switches completed their rebooting before the technical center comebacks to a normal state. In the network, where about 30 SWs depend on one TC, more than 30! alarm sequences are possible for a TC_break corresponding to the 1 to 30 $n004$ alarms reaching the supervisor.

The last problem is that the system is not diagnosable as defined in [19], or to be more precise, taking this definition will lead us to get only one class containing all the faulty situations. This is mainly due to the masking problem discussed above and to the uncertainty of temporal information. Distinct faults can result in the same sequence of alarms. For example one of the possible alarm sequences for a TC_break is $cvhs(TC)$, $cves(TC)$. But a TC_cut (where the link between the supervision center and the technical center is cut), results in exactly the same sequence of alarms. In the same way, one of the possible sequences for a TC_break is $cvhs(TC)$, $cves(TC)$, $n004(SW)$. But when a TC_cut and an SW_break occur at the same time, it can result in exactly the same sequence of alarms.

Accepting this fact means building a diagnoser able to detect a faulty situation but unable to propose more informative diagnoses. In most of the case, however, it is possible to have a more precise idea of what happened: by chance, the worse case is not the most current one and it is often possible to take profit of received alarms to discriminate between candidates.

1.4 The expert system

The problem is to analyze the stream of alarms reaching the supervision center (SC), in order to follow the evolution of the network and identify the main abnormal situations. In fact, such a system currently exists, its function being to assist the operator in charge of the supervision. It is an expert system which relies on about two hundred production rules.

For example, the rule given in figure 2 means that if a *cvhs* alarm is received at time $t1$, a *cves* alarm at time $t2$ with $t2 > t1$ and more than three $n004$ alarms are received in $[t2, t2 + 30]$ then it is probable that a TC_break occurred. This rule is clearly based on the experience of users. Another possibility explaining this sequence of alarms in theory would be the occurrence of a TC_cut on a technical center and of three SW_breaks of the switches which depend on this technical center. But this alternative has been considered unlikely by the experts, who use their experience to discard it. They have an idea of the respective delays before a technical center and a switch return to a normal state; they know that a technical center usually has between ten and thirty switches depending on it, which explains that a TC_break will generally manifest itself by more than three non-masked $n004$ alarms.

This expert system is efficient (real time) and gives satisfactory results. But the problem with such a system relying on expertise is that as soon as a new type of component is used to replace an old one, or as soon as the network structure is modified, there is no guarantee that

the expertise fits any longer. For example, if the network suddenly had only two switches under a TC, the rule would no longer fit.

```
As soon as:
   a cvhs alarm, emitted by TC, is received at time t1
and
   a cves alarm, emitted by TC, is received at time t2
with  t1<t2
and
  more than three n004 alarms emitted by the SWs
  under TC, are received, during [t2, t2+30]
do
  write "TC broke down between t1 and t2"
```

Figure 2: Rule of the expert system

2 The network model

The challenge therefore was to be as efficient as the existing expert system but to provide greater generality and flexibility to be able to adapt to technological and reconfiguration changes in the network. We decided to use model-based techniques which are recognized to be more adapted to evolutive systems than expertise-based approaches are. This section describes the formalism we used to model the network. Finally, section 2.4 shows how this model can be built by composing elementary models (in fact communicating finite state machines) stored in a library. The composition operation is based on the cross-product of finite state machines.

From a supervision point of view, the network is made up of a set of interconnected components, each of which can emit or receive messages via its ports. It was therefore very natural to choose communicating finite state machines to model this discrete-event system. In fact, when a component receives a message two cases can be distinguished: the message can affect the component state or the message is only transmitted by the component to another component. In order to take this fact into account, we have introduced two different models:

- a structural model, which represents the components of the network, their connections and their (non-)transmitting states;
- a behavioral model, which represents how each component behaves and, more precisely, how it reacts to incoming messages by changing its state and emitting messages to other components.

In the following we describe each model and how they interact.

2.1 Structural model

The structural model represents the propagation of the messages between the components of the network. The network is represented by a graph whose nodes are components and edges are the connections between these components. In our application this graph is a tree as the network is hierarchical. At this level, only the transmitting and non-transmitting states of the components matter. Edges are labeled by temporal constraints representing the time needed for a message to go from one component to another.

In order to formally describe this model we introduce the following notations:

- $SET_OF_COMPONENTS$ is the set of the network components.
- SET_OF_PORTS is the set of ports of the network.
- $DELAY$ is the set of time intervals $[t_{min}, t_{max}]$.
- *port* is a function from $SET_OF_COMPONENTS$ to $\mathcal{P}(SET_OF_PORTS)$. It relates each component to its set of ports.
- *connection* is a function from $SET_OF_COMPONENTS \times SET_OF_PORTS$ to $SET_OF_COMPONENTS \times SET_OF_PORTS \times DELAY$.
  $connection(c1, p1) = (c2, p2, d)$ means that port $p1$ of component $c1$ is connected to port $p2$ of component $c2$. Moreover if a message is emitted at time t on port $p1$, it will be received on port $p2$ at time t' where t' belongs to $[t + d1, t + d2]$ where $d = [d1, d2]$.
- *transmitter* is a boolean function from $SET_OF_COMPONENTS$ to $\{true, false\}$ indicating whether the component is in a transmitting state or not.

With these notations, a structural model is a 5-tuple:
$(SET_OF_COMPONENTS, SET_OF_PORTS, port, connection, transmitter)$

2.2 The behavioral model

The behavioral model represents the state of the components and how they react to failures. The behavioral model of each component is described by a communicating finite state machine [6, 1]. This formalism has proved to be well suited to modeling dynamic systems [17]. Each transition of the finite state machine is triggered by the reception of exactly one message and is represented by figure 3.

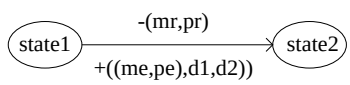


Figure 3: Transition

It means that when the component c receives the message mr on port pr at time t, it switches from $state1$ to $state2$ and sends the message me on port pe at time $t + d$ where

$d \in [d1, d2]$. d therefore represents an emission delay for the message me. The component c' which will receive the message is linked to the receiving port pe by the *connection* function. The message will be received by the component c' at time $t+d+d'$ where d' is the connection delay between c and c' (we have $connection(c, pe) = (c', pe', d')$).

Formally a communicating finite state machine is a 4-tuple (Q, Σ, δ, x_0)[2] where Q is a finite set of states, $x_0 \in Q$ is the initial state, Σ is a set of events. In our application $\Sigma = \Sigma_e \cup \Sigma_r$, where Σ_e are emitted events and Σ_r are received events.

An event is represented by a pair (m, p) where m is the type of message and p belongs to SET_OF_PORTS.

δ is the transition function mapping:

$$\delta : Q \times \Sigma_r \rightarrow Q \times (\Sigma_e \times DELAY)^*$$

Definition : internal message / external message
A message emitted from a component c to itself is an internal message; a message emitted from a component c to another component c' or an exogenous message (coming from the outside as failure events) is an external message.

Definition : active message/passive message
A message m received on port p of component c is active for this component if and only if $\exists s \in Q \mid \delta(s, (m, p)) = (s', -)$ with $s' \neq s$ or $\delta(s, (m, p)) = (s, \{(m_1, p_1, d_1) \cdots (m_n, p_n, d_n)\})$ such that $\exists i$ with $m_i \neq m$. It means that the message, when received by the component, either changes its state or triggers the emission of at least one new message.

A message m received on port p is passive for a component c if and only if it is not active. Such a message can only be transmitted by the component to another one. A passive message is a message which is passive for all the components of the network.

Definition : alarm
An external message emitted by any component directly to the supervisor or to the supervisor by way of other components which only transmit it (without emission delay and without changing its type) is referred to as an alarm. An alarm is therefore a passive message. Conversely, a passive message is generally an alarm.

2.3 Links between the two models

- A function cs associates to each node of the transmission graph an automaton which describes how the component behaves.
- A function $f : Q \rightarrow \{true, false\}$ is defined for each automaton, where Q is the set of automaton states. It indicates whether the state Q is a transmitting state or not.
- Given the current state cs of a component c, the link between the structural model and the behavioral model is made by the following constraint:
  $\forall c \in SET_OF_COMPONENTS, f(cs(c)) = transmitter(c)$.

[2] In [18, 3] this formalism is extended with actions and guard conditions as is often done for modeling communication protocols [17].

- The structural model is sufficient to model the propagation of the alarms (and the passive messages) through the network. The finite state machine representing the behavioral model of a component can thus be simplified by not taking into account its transmission role, which means that transitions triggered by the reception of passive messages will not be considered. Remember that, by definition, a passive message cannot have any effect on the component (no change of state, no emission of new messages).

2.4 Building the model by composing elementary models

To be more efficient and to make the modeling stage easier, a composition operation is provided to build the model. It is based on the cross product of the corresponding communicating finite state machines and includes the deletion of all internal messages which are useless for the supervision process.

The composition techniques used here are inspired by the synchronous composition of automata described in [19]; in addition, simplification rules are used to obtain an abstract representation of the components where some internal messages are removed. In the following the synchronous composition and the composition of communicating automata are described.

2.4.1 Definition of the synchronous product

In this section the system is assumed to consist of several distinct physical components, each of which is represented by a finite state machine $G_i = (Q_i, \Sigma_i, \delta_i, s_{0i})$. Q_i is the state space, Σ_i is the set of events, δ_i is the transition function, with no temporal constraints, and s_{0i} is the initial state of G_i.The joint operation of two or more automata corresponds to the synchronous composition procedure. Consider two discrete event systems G_1 and G_2. We denote $e_i(x)$ the active event set of G_i at state x, i.e. the set of all transitions of G_i at state x. Let $G = (Q, \Sigma, \delta, s_0)$ denote the synchronous composition of G_1 and G_2. Then

$$\begin{aligned}
\Sigma &= \Sigma_1 \cup \Sigma_2 \\
Q &= Q1 \times Q_2 \\
\delta((s_1, s_2), \sigma) &= \begin{cases} (\delta_1(s_1, \sigma), \delta_2(s_2, \sigma) \text{ if } \sigma \in e_1(s_1) \cap e_2(s_2) \\ (\delta_1(s_2, \sigma, s_1)) \text{ if } \sigma \in e_1(s_1) - \Sigma_2 \\ (s_1, \delta_2(s_2, \sigma)) \text{ if } \sigma \in e_2(s_2) - \Sigma_1 \end{cases}
\end{aligned}$$

In the following subsections, the composition of communicating automata, without delay, is presented. Two parts are distinguished : the composition of the structural models and the composition the behavioral models.

2.4.2 Composition of the structural models

Let a structural model be described by a 5-tuple:

$(SET_OF_COMPONENTS, SET_OF_PORTS, port, connection, transmitter)$
where :

$$SET_OF_COMPONENTS = \{C_1, C_2\}$$
$$SET_OF_PORTS = P_1 \cup P_2$$
$$port(C_1) = P_1 = P_{1,ext} \cup P_{1,int}$$
$$port(C_2) = P_2 = P_{2,ext} \cup P_{2,int}$$
$$connection \in (C_1 \times P_{1,int}) \rightarrow (C_2 \times P_{2,int}) \cup (C_2 \times P_{2,int}) \rightarrow (C_1 \times P_{1,int})$$
$$transmitter \in SET_OF_COMPONENTS \rightarrow \{true, false\}$$

$P_{1,int}$ (resp. $P_{2,int}$) is the set of ports of C_1 (resp C_2) which are connected to C_2 (resp. C_1), $P_{1,ext} = P_1 - P_{1,int}$ and $P_{2,ext} = P_2 - P_{2,int}$.

After composition of C_1 and C_2 in C, the model is the following 5-tuple : $(SET_OF_COMPONENTS', SET_OF_PORTS', port', connection', transmitter')$ where all the connections joining C_1 to C_2 have been taken off (with all the associated ports).

$$SET_OF_COMPONENTS' = \{C\}$$
$$SET_OF_PORTS' = P_{1,ext} \cup P_{2,ext}$$
$$port'(C) = SET_OF_PORTS'$$
$$connection' = \emptyset$$
$$transmitter'(C) = transmitter(C_1) \wedge transmitter(C_2)$$

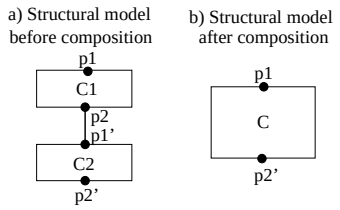


Figure 4: Structural model and composition

Let consider the model given by figure 4. Before composition, the 5-tuple is given by :

$$SET_OF_COMPONENTS = \{C_1, C_2\}$$
$$SET_OF_PORTS = \{p1, p2, p1', p2'\}$$
$$port(C_1) = \{p1, p2\} \text{ with } P_{1,int} = \{p2\}\ P_{1,ext} = \{p1\}$$
$$port(C_2) = \{p1', p2'\} \text{ with } P_{2,int} = \{p1'\}\ P_{2,ext} = \{p2'\}$$
$$connection = \{p1' \rightarrow p2, p2 \rightarrow p1'\}$$
$$transmitter \in SET_OF_COMPONENTS \rightarrow \{true, false\}$$

After composition, the ports $p2$, $p1'$ and the connections between $p2$ to $p1'$ are taken off. The resulting structural model is :

$$\begin{aligned}
&SET_OF_COMPONENTS' = \{C\}\\
&SET_OF_PORTS' = \{p1, p2'\}\\
&port'(C) = \{p1, p2'\}\\
&connection' = \emptyset\\
&transmitter' = transmitter(C_1) \wedge transmitter(C_2)
\end{aligned}$$

2.4.3 Composition of the behavioral models without delay

Let the behavioral models of component C_1 and C_2 be given by:

$$\begin{aligned}
& & C_1 &= (Q_1, \Sigma_1, \delta_1, s_{0,1})\\
& & C_2 &= (Q_2, \Sigma_2, \delta_2, s_{0,2})\\
&\text{where for } i = 1, 2 & \Sigma_i &= \Sigma_{i,e} \cup \Sigma_{i,r}\\
& & \Sigma_{i,e} &= \Sigma_{i,e,int} \cup \Sigma_{i,e,ext}\\
& & \Sigma_{i,r} &= \Sigma_{i,r,int} \cup \Sigma_{i,r,ext}
\end{aligned}$$

$\Sigma_{i,e,int}$ is the set of emitted internal messages. We have $m = (name, port) \in \Sigma_{i,e,int} \Leftrightarrow m \in \Sigma_{i,e} \wedge port \in P_{i,int}$. $\Sigma_{i,e,ext}, \Sigma_{i,r,int}$ and $\Sigma_{i,r,ext}$ are respectively the set of emitted external messages, received internal messages and received external messages and are defined in the same way as $\Sigma_{i,e,int}$.

The behavioral model of the composed model is given by :

$$\begin{aligned}
&C = (Q, \Sigma_r \cup \Sigma_e, \delta, (s_{0,1}, s_{0,2}))\\
&Q = Q_1 \times Q_2\\
&\Sigma_r = \Sigma_{1,r,ext} \cup \Sigma_{2,r,ext}\\
&\Sigma_e = \Sigma_{1,e,ext} \cup \Sigma_{2,e,ext}\\
&\delta : Q_1 \times Q_2 \times \Sigma_r \rightarrow Q_1 \times Q_2 \times (\Sigma_e)^*
\end{aligned}$$

Before defining formally δ, let us explain intuitively how it works. When an external message is received by $C1$, an internal message can be sent from $C1$ to $C2$. Receiving an internal message, $C2$ can now send another internal message to $C1$ and so on. In the following definition, $\sigma_{1,2}, \ldots, \sigma_{1,n}$ (resp. $\sigma_{2,1}, \ldots, \sigma_{2,m}$) are the internal messages sent by $C1$ to $C2$ (resp. $C2$ to $C1$). As the components alternatively send messages and as $C1$ begins, we have $m = n$ or $m = n - 1$. Moreover $s_{1,1}, \ldots, s_{1,n}$ (resp. $s_{2,1}, \ldots, s_{2,m}$) denote the states of $C1$ (resp. $C2$) after each internal message reception. To simplify, we also use the predicate $connect_1$ and $connect_2$ defined as follows : $connect_1 \in \Sigma_{1,e,int} \rightarrow \Sigma_{2,r,int}$ *with*

$connect_1(m,p) = (m,p')$ *ssi* $connection(c_1,p) = (c_2,p')$ and $connect_2 \in \Sigma_{2,e,int} \rightarrow \Sigma_{1,r,int}$ *with* $connect_2(m,p) = (m,p')$ *ssi* $connection(c_2,p) = (c_1,p')$. In order to define δ two projections are introduced : P_Q and P_E which respectively projects $Q \times (\Sigma_e^*)$ into Q and $Q \times (\Sigma_e^*)$ into (Σ_e^*).

In the following, the definition is given for the case where an external event is received by $C1$; the case where an external event is received by $C2$ could be defined in an analogeous way.

For all $\sigma_{1,1} \in \Sigma_{1,r}$, $\delta = \delta((s_{1,1}, s_{2,1}), \sigma_{1,1})$ is defined by

$$
\begin{array}{ll}
\delta = & (P_Q(\delta_1(s_{1,1},\sigma_{1,1}), s_{2,1}), P_E(\delta_1(s_{1,1},\sigma_{1,1}))) \\
\text{iff} & P_E(\delta_1(s_{1,1},\sigma_{1,1})) \subseteq \Sigma_{1,e} \\
\delta = & ((s_{1,n}, s_{2,m}), em) \\
\text{iff} & \exists(s_{1,1},\ldots,s_{1,n}) \in (Q_1)^n \ \exists(s_{2,1},\ldots,s_{2,m}) \in (Q_2)^m \\
 & \text{ with } m = n \text{ or } m = n-1 \\
 & \exists(\sigma_{1,2},\ldots,\sigma_{1,n}) \in (\Sigma_{1,r,int})^{n-1} \ \exists(\sigma_{2,1},\ldots,\sigma_{2,m}) \in \Sigma_{2,r,int}^m \\
 & \text{ such that} \\
 & \forall i \in [1,n] \ P_Q(\delta_1(s_{1,i}),\sigma_{1,i}) = s_{1,i+1} \\
 & \forall i \in [1,n] \text{ if } i \neq n \text{ or } m = n \\
 & \quad P_E(\delta_1(s_{1,i},\sigma_{1,i})) \cap \Sigma_{1,e,int} = \{\sigma\} \text{ and } connect_1(\sigma) = \sigma_{2,i} \\
 & \forall j \in [1,m] \ P_Q(\delta_1(s_{2,j},\sigma_{2,j})) = s_{2,j+1} \\
 & \forall j \in [1,m] \text{ if } j \neq m \text{ or } m = n-1 \\
 & \quad P_E(\delta_1(s_{2,j}),\sigma_{2,j}) \cap \Sigma_{2,e,int} = \{\sigma\} \text{ and } connect_2(\sigma) = \sigma_{1,j+1} \\
 & \text{if } m = n \ P_E(\delta_2(s_{2,m}),\sigma_{2,m}) \cap \Sigma_{2,e,int} = \emptyset \\
 & \text{if } m = n-1 \ P_E(\delta_1(s_{1,n})\sigma_{1,m}) \cap \Sigma_{1,e,int} = \emptyset
\end{array}
$$

In this definition, $em = (\bigcup_{i=1}^n P_E(\delta_1(s_{1,i},\sigma_{1,i})) - \Sigma_{1,e,int}) \cup (\bigcup_{i=1}^m P_E(\delta_2(s_{2,i},\sigma_{2,i})) - \Sigma_{2,e,int})$.

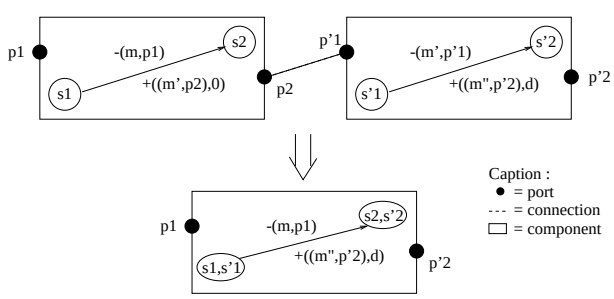


Figure 5: Composition

Let us consider an example of two components $c1$ and $c2$, whose structural and behavioral models are given by figure 5. A single external event can be received: $(m, p1)$. Therefore the composed model has a single transition : $\delta((s1, s1'), (m, p1)) = ((s2, s2'), \{(m'', p2')\})$ as $\delta_1(s1, (m, p1)) = (s2, \{(m', p2)\})$ and $\{(m', p2)\} \cap \Sigma_{1,e,int} = \{(m', p2)\}$ and $connect\{(m', p2)\} = \{(m', p1')\}$ and $\delta_2(s1', (m', p1')) = (s2', \{(m'', p2')\})$ and $\{(m'', p2')\} \cap \Sigma_{2,e,int} = \emptyset$.

2.4.4 Composition with delay

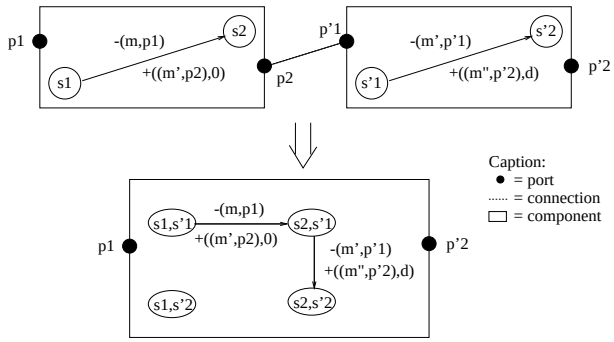


Figure 6: Composition with delay

Let us consider the case where messages (for instance m' in our example) are sent with delays and illustrate what happens on the example. Concerning the structural model, if the message m' was sent with a delay, $p2$ and $p'1$ would be kept as internal ports of $c1 \cup c2$. Concerning the behavioral model, the message m' can no longer be removed as it was the case in 5. The message is then kept as shown by figure 6.

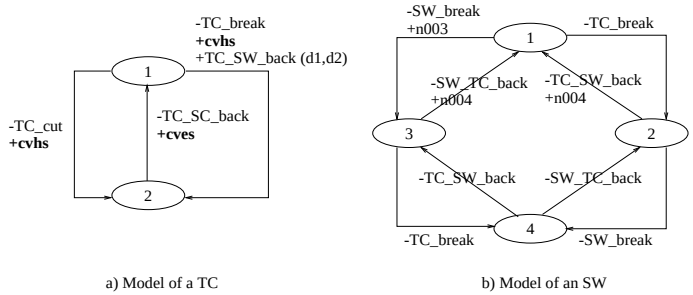


Figure 7: Elementary models of a TC and an SW

The last problem is to know which components have to be composed. Following [20], we rely on a global model of the system; the composition then goes on until model of the system has been obtained; the structural model contains a single component. In section 5, we rely on a generic model of the system which results from the composition of all the components of a branch of the network.

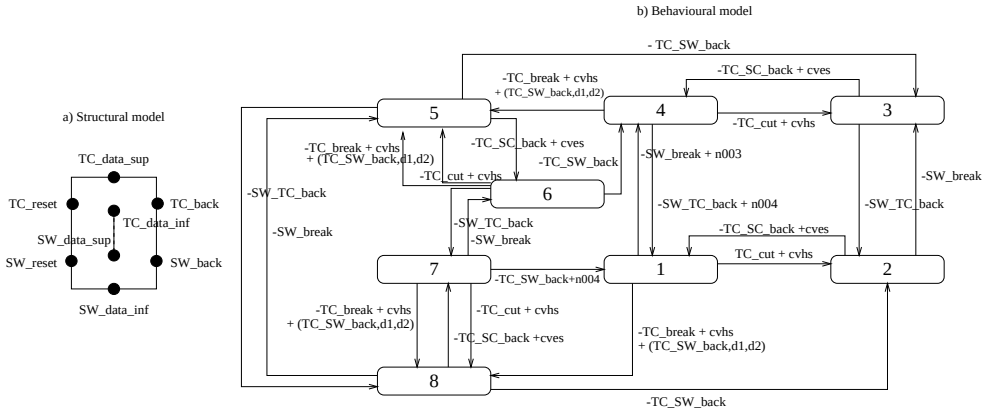


Figure 8: Model resulting from the composition of a TC and an SW (to simplify, SW_cut is not represented)

Let us illustrate this section by giving the model corresponding to a technical center TC connected to a switch SW. Their elementary structural and behavioral models are given in figure 7. The global model is obtained by composition and is given in figure 8. For sake of clarity, ports are not specified in the behavioral model. The following table associates the messages and the ports where they are emitted or received. The only internal message, TC_SW_back, is emitted on port TC_data_inf with the delay $(d1, d2)$.

TC_reset	TC_cut, TC_break
SW_reset	SW_break
TC_back	TC_SC_back
SW_back	SW_TC_back
SW_data_sup	TC_SW_back
TC_data_inf	TC_SW_back
TC_data_sup	cvhs, cves, n004, n003

3 Diagnoser

3.1 Reasons for a diagnoser approach

Using model-based reasoning techniques allows the supervision knowledge to be acquired in a more systematic, robust and evolutive way than expertise. Nevertheless, the combina-

torial cost of the model-based techniques, when used on-line, rarely satisfies the real-time constraints of supervision systems. For this reason we decided to use the model off-line and to adopt the diagnoser approach as proposed by [21, 20]. In this approach, the discrete event model of the system to be diagnosed is transformed into a finite state automaton, called a diagnoser. The transition of the automaton corresponds to observable events; the state of this automaton contains information on the system state and on the failures that have occurred; the current state of the diagnoser changes according to occurring observable events. This approach verifies both efficiency and genericity. The expensive combinatorial part is the building of the diagnoser which is done off-line. The genericity comes from the use of a model from which the diagnoser is built automatically.

This section continues with a presentation of the original diagnoser approach. The following section explains the difficulties that were specific to our application and which forced us to extend it. As it relied on a classical finite-state machine formalism, we had first to extend it to communicating finite-state machines. We had also to cope with other specificities, including the existence of intermittent faults and the importance of temporal information. Section 5 continues with a discussion of model size for a system such as a telecommunication network and presents what we called the generic diagnoser to solve this problem.

3.2 The original diagnoser approach

In the diagnoser approach as described in [21], the system is assumed to consist of several distinct physical components, each of which is represented by a finite state machine $G_i = (Q_i, \Sigma_i, \delta_i, x_{0i})$. Q_i is the state space, Σ_i is the set of events, δ_i is the transition function, with no temporal constraints, and x_{0i} is the initial state of G_i. In the first step, a model $G = (Q, \Sigma, \delta, x_0)$ of the whole system is built using a composition operation on automata.

Some of the events in Σ are observable (Σ_o), their occurrence can be observed. The rest, which are unobservable (Σ_{uo}), includes failures. The hypotheses are that no G_i has an unobservable cycle (i.e. a cycle with only unobservable events) and that failures are permanent. The set of failure events ($\Sigma_f \subset \Sigma_{uo}$) is also partitioned into disjoint sets corresponding to different failure types:

$$\Sigma_f = F_1 \cup F_2 \cup \ldots \cup F_m$$

The aim of the diagnosis is to make inferences about past occurrences of failure types on the basis of the observed events. In order to solve this problem the system model is directly converted into a "diagnoser".

The diagnoser, as defined in [21], is a deterministic finite state machine $D = (Q_d, \Sigma_d, \delta_d, x_{0d})$ where the set of events Σ_d is the set of observable events of the system Σ_o ($\Sigma_d = \Sigma_o$). Each state of the diagnoser contains the possible states of the system and, for each state, a label includes all the failures that must have occurred in order to reach this state. For example, when the diagnoser is in the state $q_d = \{(x_1, l_1), (x_2, l_2), \ldots, (x_n, l_n)\}$ it means that the

system is in one of the states $x_1, x_2, \ldots, x_n$. Moreover, if the system is in the state x_i, the failures included in l_i must have occurred. A label l is as follows:

- $l = \{N\}$: no failure what so ever occurred.
- $l = \{A\}$: some failures may or may not have occurred. If so, none can be identified (A means ambiguous).
- $l = \{F_{i_1}, F_{i_2}, \ldots, F_{i_k}\}$: at least one failure for each of the $F_{i_1}, F_{i_2}, \ldots, F_{i_k}$ has occurred.

The system is assumed to be normal to start with and hence the initial state of the diagnoser is $(x_0, \{N\})$.

Let us briefly see how to build δ_d, for details see [21]. It is a recursive process. Considering a state of the diagnoser q_1, for each possible observed event σ a new state of the diagnoser $q_2 = \delta(q_1, \sigma)$ is computed following a three step process:

- For each possible state of the system x in q_1, get all the states of the system which can be reached from x, having observed σ, by $S(x, \sigma) = \{\delta(x, s\sigma)\}$ where $s \in \Sigma_{uo}^*$. Σ_{uo}^* denotes the Kleene closure of the set Σ_{uo}. It contains all the words built from Σ_{uo}.
- Let $x' \in S(x, \sigma)$ with $\delta(x, s\sigma) = x'$. Propagate the label l associated with x to the label l' associated with x' according to the following rules[3]:
  - if $l = \{N\}$ and s contains no failure events, the label $l' = \{N\}$
  - if $l = \{A\}$ and s contains no failure events, the label $l' = \{A\}$
  - if $l = \{N\}$ or $l = \{A\}$ and s contains failure events from $\{F_{i_1}, \ldots, F_{i_m}\}$ then $l' = \{F_{i_1}, \ldots, F_{i_m}\}$
  - if $l = \{F_{i_1}, \ldots, F_{i_m}\}$ and s contains failure events from$\{F_{j_1}, \ldots, F_{j_k}\}$ then $l' = \{F_{i_1}, \ldots, F_{i_m}\} \cup \{F_{j_1}, \ldots, F_{j_k}\}$
- Let q_2 be the set of all (x', l') pairs computed following steps 1 and 2 above, for each (x, l) in q_1 and with a given σ. Replace all $(x', l'), (x', l'') \in q_2$ by $(x', l' \cap l'')$ if $l' \cap l'' \neq \emptyset$ or by (x', A) otherwise. That is to say, if the same estimated state x' appears more than once in q_2 with different labels, we associate all common components of these labels with x'. If there is no common components we attach the ambiguous label A to x.

For example, consider the system whose finite state machine is given in figure 9. The events oi are observable events, whereas Fi are failures. The diagnoser of such a system is given in figure 10. Other examples of diagnosers can be found in [21].

The diagnoser is exploited on line as follows. Given a sequence of observable events and a current state, the state which can be reached by transiting along the path of observables gives the set of all the possible states of the system associated to a set of failures explaining

[3]The treatment of the ambiguous symbol has been lightly simplified

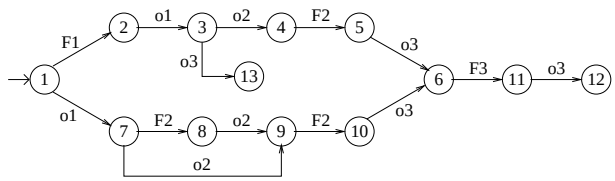


Figure 9: The finite state machine describing the system behavior

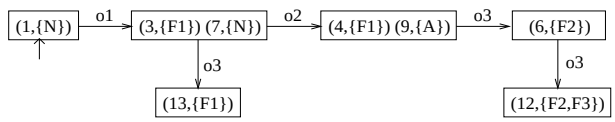


Figure 10: The diagnoser

the observations. Being in the initial state $(1, \{N\})$ and observing the sequence $o1, o3$, the diagnoser reaches the state $(13, \{F1\})$. Therefore we know that the system is in state 13 and that one of the failures of type $F1$ has occurred.

In order to deal with multiple failures, [20] proposes a new definition in which the ambigous symbol is taken off. A state of the diagnoser can include a same model state many times with different labels. On the example, we can go from state 7 to state 9 by two ways which are both kept instead of merging them within the ambigous symbol. The transition of the diagnoser labeled by $o2$ and starting from state $((3, \{F_1\})(7, \{N\}))$ goes now in state $((4, \{F_1\})(9, \{N\})(9, \{F_2\}))$ instead of $((4, \{F_1\})(9, \{A\}))$ as before. The new diagnoser is given by figure 11. This new definition of diagnoser has been introduced to determine, in the case of multiple failures, if the system is diagnosable. The formal definition of the new diagnoser is given in [20].

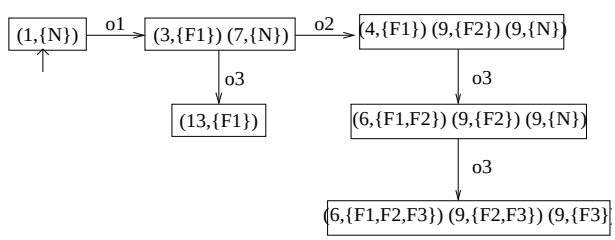


Figure 11: The diagnoser without the ambigous symbol

4 Extending the diagnoser approach to deal with our application

The diagnoser approach is attractive for our application for two reasons. The diagnostic is realized on-line by following the state of a finite state machine according to observed events. Therefore it is efficient. The diagnoser is built off line from the global model of the system which is itself built from models of elementary components. When the system is modified, the elementary models are updated; consequently a new model and a new diagnoser are built, which gives this approach genericity. Unfortunately, the diagnoser approach as proposed by [21] concerns synchronous finite state machines. In our application we have chosen to use temporal communicating finite state machines. This formalism is well suited to modeling telecommunication networks or protocols [17], where components exchange messages. Moreover we need to be able to express temporal constraints on the delays (emission and transmission delays of messages). It was thus richer formalism as well as some specificities of our application which forced us to extend the diagnoser approach.

As in [20], we rely on the global model of the system to be surveyed[4]. This global model may have been built by composing elementary models, as explained in subsection 2.4. In this model, three kinds of messages exist: internal messages, alarms and failure messages. The failure messages include the messages corresponding to the occurrence of a failure but also the comeback messages. Internal messages and failure messages are not observable. Emitted events can therefore be internal events or alarms. Received events are failure or internal events. The model of the system is a 4-tuple (Q, Σ, δ, x_0) where $\Sigma = \Sigma_e \cup \Sigma_r$, $\Sigma_r = \Sigma_{ir} \cup \Sigma_f$, $\Sigma_e = \Sigma_{ie} \cup \Sigma_a$ with Σ_{ir} the set of received internal events, Σ_f the set of failures, Σ_{ie} the set of emitted internal events, Σ_a the set of emitted alarms. The set of observables Σ_o is defined by $\Sigma_o = \Sigma_a$ and the set of unobservables Σ_{uo} as $\Sigma_{ie} \cup \Sigma_{ir} \cup \Sigma_f$. The supervision problem consists in finding events of Σ_f explaining some observable events of Σ_o.

In the following, we first informally introduce the extension of the diagnoser we have proposed. Each difficulty is given and the proposed solution is explained and illustrated. This is followed by a formal definition of the extended diagnoser[5] the algorithm used to build it from the model of the system (see 4.5 and 4.6).

The main extensions described in the following subsections are as follows:

- The diagnoser has to deal with a *communicating* finite state machine, so we have to deal with exchanged messages.
- We need to take into account the emitted events especially when they are emitted with delays. This is why conditions on these internal events have been added to the states of the extended diagnoser.

[4]This point can be problematic. We are currently working on solutions to solve this problem such as generic diagnosers presented in section 5 and communicating diagnosers mentioned in section 8.

[5]The term "extended diagnoser" used throughout this paper refers to the extension of the basic diagnoser ([20]) presented in this paper. It does not refer to the data structure also named "extended diagnoser" used in [11].

- We need to keep information about failures that may have occurred. It is the reason why we have discarded the ambiguous symbol A from the state label, which becomes a list of possible failures.
- We need to take into account non-permanent failures. The model can therefore contain cycles, which is why we have added output messages to the transitions.
- We need to take into account temporal constraints. It is the reason why we have associated a date to each event and why we have associated temporal constraints to the dates of the possible faults.

4.1 Dealing with a communicating finite state machine (CFSM)

In the following two sections, for reasons of clarity, we suppose that the model of the system does not include temporal constraints; they will be introduced in 4.4.

4.1.1 A simplified case

Let us first consider the case where there are no internal messages. In this case, messages can only be alarms and failure messages. Alarms are emitted events and failure messages are received events. A state of the diagnoser is $q = (x, \{l\})$ where x is a single state of the behavioral model of the system and l a set of failures. To build the transitions leaving state $q = (x, \{l\})$ we look for all words so where $s = s_1 \ldots s_n$, $s_i \in \Sigma_{uo}$ and $o \in \Sigma_o$, which allow the automaton of the system to go from x to x' by receiving s and then sending o. For each word so the transition $\delta((x, \{l\}), o) = (x', \{l, s_1, \ldots, s_n\})$ is added.

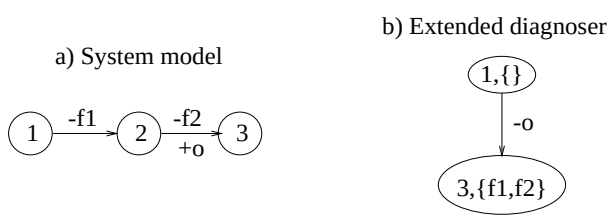


Figure 12: Building the extended diagnoser from a CFSM

For example, let us consider the system whose model is given in figure 12.a. From state 1, the only observable event is o. o can lead to state 3 on condition of failures $f1$ and $f2$. A transition is thus moving from $(1, \{\})$ to $(3, \{f1, f2\})$, as shown in figure 12.b.

4.1.2 Dealing with internal events

If we now consider internal events, both received and emitted events can be internal events. As internal events are not observable events, when we look for all received words so leaving x, with $s \in \Sigma_{uo}^*$, s can include internal events. But an internal event should not be received if it has not been emitted. Therefore we store all internal events which have been emitted and not yet received, which make it possible to check received messages against expected ones and to eliminate some of the impossible alternatives. We add to each state of the extended diagnoser a list of such internal events. A state of the extended diagnoser is now $\{(x, l, i)\}$ where i is a list of internal events. To build the extended diagnoser from state $q = \{(x, l, i)\}$, we look in the model of the system for all words $s_1 \ldots s_n o$ enabling a transition from x to x' and where

- $s_i \in \Sigma_f$ (it is a received message),
- or $s_i \in \Sigma_{ir}$ and $s_i \in i$ (s_i is an internal event which was expected and has now been received),
- or $s_i \in \Sigma_{ie}$: s_i is added as an expected event,
- and the observable $o \in \Sigma_o$ is emitted by the last transition.

The new state is $q' = \{(x', l', i')\}$ with $l' = l \cup \{s_i | s_i \in \Sigma_f\}$ and $i' = i - \{s_i | s_i \in \Sigma_{ir} \wedge s_i \in i\} \cup \{s_i | s_i \in \Sigma_{ie} \wedge s_i \notin i\}$. The list of expected internal events is updated by deleting the received ones and adding the emitted ones.

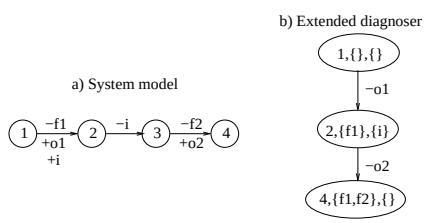


Figure 13: The extended diagnoser with internal events

For example, let us consider the system model in figure 13. $o1$ enables a transition from state 1 to state 2 but with the emission of an internal event i. Therefore we add to the diagnoser the state $(2, \{f1\}, \{i\})$, meaning that i has been sent but has not been received. From the extended diagnoser state $(2, \{f1\}, \{i\})$, we can go to state $(4, \{f1, f2\}, \{\})$, where i is taken out of the internal list as it has been received.

In the following, we will consider that all internal events are emitted concurrently to observable events. This is the case in our application and a simpler algorithm is therefore possible.

4.2 Keeping information about failures that may have occurred

Diagnosability of a system with respect to a disjoint set of failure types is usually defined as the fact that the occurence of any failure types can be identified using a bounded number of observations following the occurence of that failure. A formal definition can be found in [20] which corresponds to the following : let s be any trace generated by the system that ends in a failure state of type Fi (let us recall that the set of failures Σ_f is partitioned in a set of disjoint types of failures $F_1 \cup F_2 \cup \ldots \cup F_m$) and let t be any sufficiently long continuation of s. The condition requires that every trace belonging to the language of G and having the same observable trace than st should contain the occurrence of a failure of the type Fi with a finite delay. It then means that every failure type event leads to a sequence of observations distinct enough to enable unique identification of the failure type with a finite delay.

Due to the masking process, the system we have considered is clearly not diagnosable as defined in [20] or, to be more precise, taking this definition will force us to consider only one type of fault containing all the faults.

To illustrate this, consider the figure 14 which gives, for each set of failures possibly occurring on a TC and its two SWs, the set of alarm sequences possibly received by the supervision center. The TC comeback and the SW comebacks can happen in any order due to the uncertainty of temporal delays. Each sequence corresponds to the way these failure events interleave. It is easy to see that there are no sequence of alarms discriminating these failures.

Another way of illustrating this fact is by looking at the automata given in figure 8. For the two faults TC_break and TC_cut, two traces can easily be exhibited corresponding to the same sequence of alarms $\{cvhs, cves\}$: the one going from state 1 to state 1 though the states 8 and 2 and the one going from state 1 to state 1 though the state 2. Consequently, following [19], TC_break and TC_cut should belong to the same type of failures. In fact, for our system, using this definition means the existence of exactly one class containing all the failures.

Accepting this fact means building a diagnoser able to detect a faulty situation but unable to propose more informative diagnoses. In most of the case, however, it is possible to have a more precise idea of what happened: by chance, the worse case is not the most current one and it is often possible to take profit of received alarms to discriminate between candidates. In our specific case, due to the masking effect, the same observable sequence can often be explained by two distinct failures, but it indicates the occurence of interacting faults which are interesting to identify. For example, the sequence $cvhs, cves, n004(SW1)$ does not discriminate between a TC_break or a TC_cut (both can produce this sequence of alarms); however, it indicates that the TC_cut has to be followed by a $SW_break(SW1)$ occurred before the TC come_back which masked the $n003$ alarm. In a monitoring context, we are interested in tracking all the possible histories of the system, even if we know that some failures cannot be identified for sure.

We propose then to use a weak definition of diagnosability which is the following : A system is said to be weakly diagnosable with respect to a given partition of Σ_f, $\Sigma_f = F_1 \cup F_2 \cup \ldots \cup F_m$, iff the following condition is satisfied. Let $s1$ be any trace generated by

the system that ends in a faulty state $f1$ of type Fi and let $t1$ be any continuation of $s1$. The condition requires that, for any fault $f2$ belonging to the same class Fi, there exists a trace $s2$ ending in the faulty state $f2$ and a continuation $t2$ such that $s1t1$ and $s2t2$ have the same observable trace and correspond exactly to the same occurrences of failures : same types in the same order. It then means that it is impossible to discriminate the failures of a same type even by looking at the failure histories explaining a sequence of observations. This weak diagnosability defines then equivalence classes with respect to what is observable. As soon as two failures do not belong to the same class, it is worthy to differentiate them for they can correspond to different failure histories of the system.

Failures having happened	Sequences of received alarms
TC_break (and its TC_SC_back)	cvhs, cves cvhs, cves, n004(SW1) cvhs, cves, n004(SW2) cvhs, cves, n004(SW1), n004(SW2) cvhs, cves, n004(SW2), n004(SW1)
TC_cut (and its TC_SC_back)	cvhs, cves
TC_cut followed by SW_break(SW1)(*)	cvhs, cves cvhs, cves, n004(SW1)
TC_cut followed by SW_break(SW2)(*)	cvhs, cves cvhs, cves, n004(SW2)
TC_cut followed by SW_break(SW1) and by SW_break(SW2)(*)	cvhs, cves cvhs, cves, n004(SW1) cvhs, cves, n004(SW2) cvhs, cves, n004(SW1), n004(SW2) cvhs, cves, n004(SW2), n004(SW1)

Figure 14: Interleaved TC and SW failures and the corresponding alarm sequences - (*) : The SW breakdowns (SW_break) are supposed to happen before the TC comeback (TC_SC_back) but the TC comeback and the SW comebacks (SW_TC_back) can happen in any order.

The main consequence of keeping information about all the possible failures explaining the observations is that the use of the ambiguous symbol as proposed in the basic diagnoser by [20] has to be given up. We use a solution similar to the one proposed in [20] to deal with multiple faults (see figure11 in section 3.2) where each possible faults explaining a state are kept in the diagnoser state. The only difference is that, instead of having a deterministic automaton, the extended diagnoser is a non deterministic automaton where each diagnoser state corresponds to one of the alternative.

Let us consider the system whose model is given in figure 15.a (using the finite state machine formalism) and in figure 15.c (using the communicating finite state machine formalism). From state 1, the only observable event is $o1$. $o1$ can lead to state 2 with the failure $f1$ or with the failure $f2$. With the diagnoser approach without ambiguous symbol

the label of the new state is $((2, \{f1\})(2, \{f2\}))$ as shown in figure 15.b. In our extended diagnoser, (figure 15.d), two transitions are added, one leading to the state $(2, \{f1\})$ and the other to the state $(2, \{f2\})$.

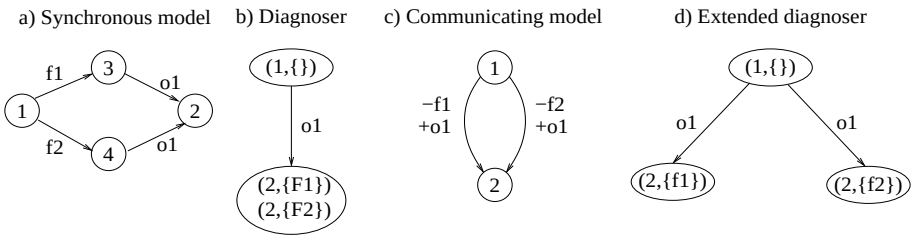


Figure 15: Building a non deterministic diagnoser

4.3 Taking into account non-permanent failures

In the initial diagnoser approach, failures are assumed to be permanent and the finite state machine which models the system is assumed to be without any unobservable cycles. This is not the case in our application where, due to recovery procedures, breakdowns are followed by "comebacks" which create cycles in the model. If this fact is ignored the extended diagnoser as presented above, will grow indefinitely (a breakdown and its comeback can appear infinitely in a label). To solve this problem, we have introduced the notion of minimal comeback set and have added output messages to the extended diagnoser. Definition : Minimal ComeBack Set

When a failure f occurs, the system changes according to this failure. As breakdowns are not permanent, thanks to the existence of recovery procedures, the failure disappears after a while. The minimal comeback set (MCBS) is the minimal set of events constituting a cycle; it makes the system come back or return to an already visited state. An MCBS can be associated to each failure. Examples of failures and their MCBS are given in the following table.

Failure	MCBS
TC_break	TC_break, TC_SC_back, TC_SW_back
TC_cut	TC_cut, TC_SC_back

When building the extended diagnoser, if an MCBS is recognized, it is deleted from the state label and added to the transition as an output message.

A transition of the extended diagnoser is triggered by the reception of an observable event but can also output a set of events. The extended diagnoser itself becomes a communicating finite state machine.

For example, let us consider the model in figure 16.a and look at the transition leaving the state $(2, \{TC_cut\})$ of the extended diagnoser. In the model, from state 2 the only observable is *cves*, which is due to the event TC_back. With the initial construction of

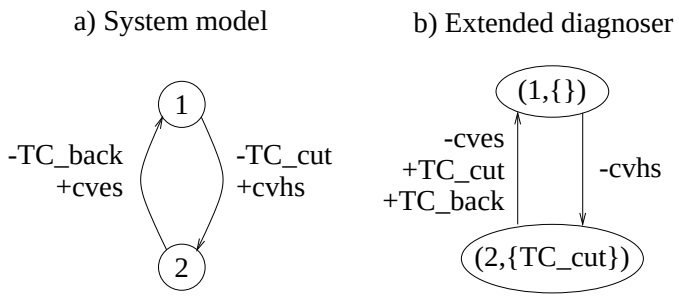


Figure 16: The extended diagnoser with output messages

the extended diagnoser, we should have the state $(1, \{TC_cut, TC_back\})$ in the extended diagnoser. This state itself could be expanded after a *cvhs* followed by a *cves* transition in a $(1, \{TC_cut, TC_back, TC_cut,$
$TC_back\})$ state and so on. The set $\{TC_cut, TC_back\}$ is recognized to be an MCBS. The state $(1, \{TC_cut, TC_back\})$ is considered as equivalent to the state $(1, \{\})$ after having added to the transition going from state $(2, TC_cut)$ to $(1, \{\})$ the output messages TC_cut and TC_back, as shown in figure 16.b.

4.4 Taking into account temporal data

The model used in the diagnoser approach, as initially proposed, has no temporal information. We have seen that it is necessary to integrate temporal constraints in order to model our network. For example, one of the sequences of events explaining the sequence of alarms $(cvhs, t1)(cves, t2)(n004, t3)$ is the following one:
$(TC_break, t1), (TC_back, t2), (SW_back(SW1), t3)$. It is a case of TC_break: TC breaks down at time $t1$, explaining *cvhs*; TC comes back at time $t2$, explaining *cves*; $SW1$ and $SW2$ begin to reboot at the same time $t1$. $SW1$ comes back to a normal state at time $t3$, explaining $n004$. $SW2$ has not come back yet, explaining that its $n004$ alarm has not yet been received. As $SW2$ takes time $d \in [d1, d2]$ to reboot, if at time $t1 + d2$ no $n004$ alarm has been received (and no other alarm indicates that a masking has occurred), this hypothesis can be eliminated.

The diagnoser is then extended by dating the events and keeping temporal constraints on these events.

- We associate a date to each observable event. $-cvhs(t1)$ means that the message *cvhs* is received at time $t1$.
- We associate a date and, whenever possible, temporal constraints to each unobservable event. $(TC_cut, t1)$ means that TC_cut occurs at time $t1$. $(TC_back, u1)$ with

$t1 < u1 < t2$ means that TC_back occurs at time $u1$ which verifies the constraint $t1 < u1 < t2$.

- In the same way, we associate a date of occurrence to the internal messages of the internal list and keep the temporal constraints known on them. $(SW_back, t1, t2)$ means that the internal message SW_back can only occur in the time interval $[t1, t2]$.
- We take advantage of the following two rules satisfied by the system model and add the corresponding temporal constraints:
  - Each event on the same transition occurs at the same time.
  - Events of two consecutive transitions are subject to a precedence constraint.

Finally, a state of the extended diagnoser is of the form $(x, l, i, \mathcal{C})$ where x is a system state, l a set of dated failures, i is a list of dated internal messages and $\mathcal{C}$ a set of temporal constraints. A transition is triggered by the reception of a dated observable event and may output a set of dated events associated with temporal constraints.

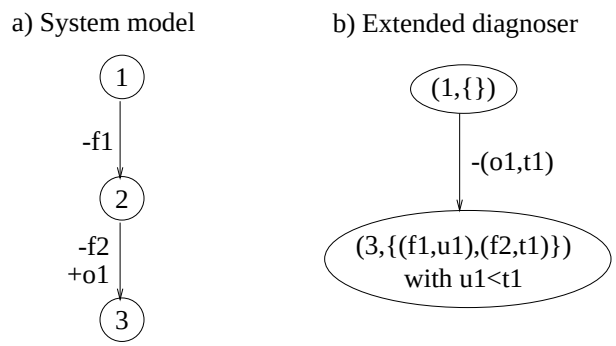


Figure 17: The extended diagnoser with temporal constraints

For example, let us consider the system shown in figure 17.a. From state 1, the only possible observable event is $o1$. Let us suppose that it occurs at time $t1$. Therefore $f2$ also occurs at time $t1$ and $f1$ at a time $u1$ which verifies the constraint $u1 < t1$. The extended diagnoser is given in figure 17.b.

Finally, to build the extended diagnoser from state $(x, l, i, \mathcal{C})$ we look at all the transitions leaving state x in the system model. Before meeting the observable event, we can encounter external events and internal events included in i. Let us suppose that we arrive at state x' through transitions $tr_1, tr_2, \ldots, tr_n$. A transition from $(x, l, i, \mathcal{C})$ to $(x', l', i', \mathcal{C}')$ is then added to the extended diagnoser, with $i' = i - i_1 + i_2$, where i_1 is the set of internal events received through $tr_1, tr_2, \ldots, tr_n$ and i_2 the set of internal events emitted through $tr_1, tr_2, \ldots, tr_n$. $l' = l + l_2 - l_1$ where l_2 is the set of all failure messages received through $tr_1, tr_2, \ldots, tr_n$ and

l_1 is the set of MCBSs belonging to $l + l_2$. The transition from $(x, l, i, \mathcal{C})$ to $(x', l', i', \mathcal{C}')$ is labeled by $-(o, t)$ $+l_1$. Moreover, constraints relative to the date of l_1 are added to $\mathcal{C}$. Figure 18 gives an example of a system with temporal constraints and of its extended diagnoser. We assume that the MCBS of $f1$ is $\{f2\}$.

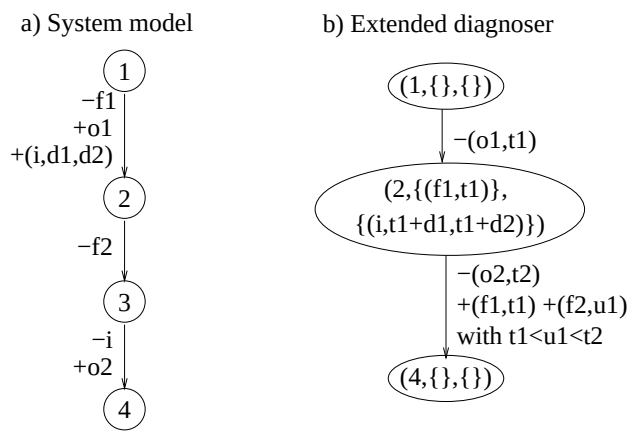


Figure 18: The extended diagnoser with internal events

4.5 Definition of the diagnoser

Before defining the extended diagnoser, let us recall the definition of the behavioral model of the network. It is a deterministic communicating finite state machine (Q, Σ, δ, x_0) where Q is a finite set of states, $x_0 \in Q$ is the initial state, Σ is a set of events. In our application $\Sigma = \Sigma_e \cup \Sigma_r$ where Σ_e are emitted events and Σ_r are received events. Moreover $\Sigma_e = \Sigma_{ie} \cup \Sigma_a$ and $\Sigma_r = \Sigma_{ir} \cup \Sigma_f$, where Σ_{ie} are emitted internal events, Σ_{ir} received internal events, Σ_f failures, and Σ_a alarms. δ is a function defined by:

$$\delta : Q \times \Sigma_r \rightarrow Q \times (\Sigma_e \times DELAY)^*$$

A diagnoser is a non deterministic communicating finite state machine $D = (Q_d, \Sigma_d, \delta_d, q_{0d})$. $\Sigma_d = \Sigma_{dr} \cup \Sigma_{de}$ where $\Sigma_{dr} = \Sigma_a$ is the set of alarms. They are the only observable events of the system, this set is called Σ_{do} in the following (Σ_{do}=$\Sigma_{dr} = \Sigma_a$). Σ_{de} is the set of MCBS.

A state of the diagnoser is $q_d = (x_s, l, i, \mathcal{C})$ where

- $x_s \in Q$ is a single state of the behavioral model of the system.

- $l = \{(f_1, t_{f_1}) \dots (f_m, t_{f_m})\}$ with $f_i \in \Sigma_f$, l is a label describing the failure events that have occurred in order to reach the state x_s. t_{f_i} is the occurrence date of the failure f_i.
- $i = \{(i_1, t_{i_{11}}, t_{i_{12}})) \dots (i_n, t_{i_{n1}}, t_{i_{n2}})\}$ with $i_j \in \Sigma_{ie}$. i is a list of internal messages including all the internal messages that have been emitted and have not yet been received. Each internal message i_j is associated with two dates $t_{i_{j1}}$ and $t_{i_{j2}}$, meaning that the message must be received between $t_{i_{j1}}$ and $t_{i_{j2}}$.
- $\mathcal{C}$ is a set of temporal constraints; a constraint looks like $t_1 < t_2$ and deals with $\{t_{f_1}, \dots, t_{f_m}, t_{i_{11}}, t_{i_{12}}, \dots t_{i_{n2}}\}$.

δ_d is a relation defined by $\delta_d : Q_d \times \Sigma_{dr} \to Q_d \times (\Sigma_{de})^*$.

4.6 Building the diagnoser

Before building the diagnoser, we first extend δ to a sequence of events.

$$\left.\begin{array}{l}\delta(x, mr_1) = (x_1, me_1)\\ \delta(x_1, mr_2) = (x_2, me_2)\end{array}\right\} \Rightarrow \delta(x, mr_1 mr_2) = (x_2, [me_1, me_2])$$

where $x, x_1, x_2 \in Q$, $mr_1, mr_2 \in \Sigma_r$, $me_1, me_2 \in (\Sigma_e \times DELAY)^*$ and $[me_1, me_2]$ denotes a sequence of dated events (the events of me_1 must occur before the events of me_2).

With the extended transition, we can now build the diagnoser recursively. The following shows how to build all the transitions $\delta_d(q_1, \sigma)$ for a state $q_1 = (x_1, l_1, i_1, \mathcal{C}_1)$ with $l_1 = \{(f_1, t_{f_1}), \dots, (f_m, t_{f_m})\}$ and $i_1 = \{(i_{11}, t_{i_{11}}, t_{i_{12}}), \dots, (i_{1n}, t_{i_{n1}}, t_{i_{n2}})\}$. For each possible observed event $\sigma \in \Sigma_{do}$ received at time t, the new states of the diagnoser q_2 are computed as follows:

- From the model of the system, we compute the set S of 4-tuple $(x_2, mr, i, \mathcal{C})$ where x_2 can be reached from x_1 by receiving the unobservable events mr and then emitting the internal events i and the observable event σ, with the constraints $\mathcal{C}$.

  $S(q_1, \sigma) = \{(x_2, mr, i, \mathcal{C})$ such that
  $\delta(x_1, mr_1 \dots mr_u) = (x_2, [(i_1, d_{i_{11}}, d_{i_{12}}), \dots (i_v, d_{i_{v1}}, d_{i_{v2}}), \sigma])\}$
  where
  $mr = \{(mr_1, t_{mr_1}), \dots, (mr_u, t_{mr_u})\}$ with $mr_i \in \Sigma_f \cup \{i_{11}, \dots i_{1n}\}$
  $i = \{(i_1, t_{i_{11}}, t_{i_{12}}), \dots, (i_v, t_{i_{v1}}, t_{i_{v2}})\}$ with $i_j \in \Sigma_{ie}$
  $\sigma \in \Sigma_{do}$ and σ and i are emitted on the last transition
  $\mathcal{C} = \{t_{mr_1} \leq \dots \leq t_{mr_{u-1}} \leq t, t_{mr_u} = t, \forall j t_{i_{j1}} = t + d_{i_{j1}}, t_{i_{j2}} = t + d_{i_{j2}}\}$

- For all $(x_2, mr, i, \mathcal{C}) \in S(q_1, \sigma)$ we add to the diagnoser the transition $(q_1, \sigma) \rightarrow \{(x_2, l_2, i_2, \mathcal{C}_2), me\}$ with

$$\begin{aligned} i_2 &= (i_1 - mr) \cup i \\ me &= MCBS(l_1 \cup mr) \\ l_2 &= l_1 \cup (mr - \Sigma_{ir}) - me \\ \mathcal{C}_2 &= \mathcal{C}_1 \cup \mathcal{C} \end{aligned}$$

  We add to the internal list all the internal messages which have been emitted on the way from x_1 to x_2, and we take out all the internal messages which have been received. $l_1 \cup l_2$ is the set of occurred failures. me is the set of MCBSs included in $l_1 \cup l_2$; it is added as output messages on the transition. The set of failure messages kept with the state x_2 is the set of received events which are not internal messages and which are not included in the set of MCBSs emitted.

4.7 Example

Let us consider a system composed of a technical center TC and a switch SW. Its model is given in figure 8. We assume that at the beginning the system is in its initial state with no failure (in the example the initial state is state 1). Therefore the initial state of the diagnoser is state $(1, \{\}, \{\})$ meaning that the system is in state 1, that no failure and no internal event have occurred.

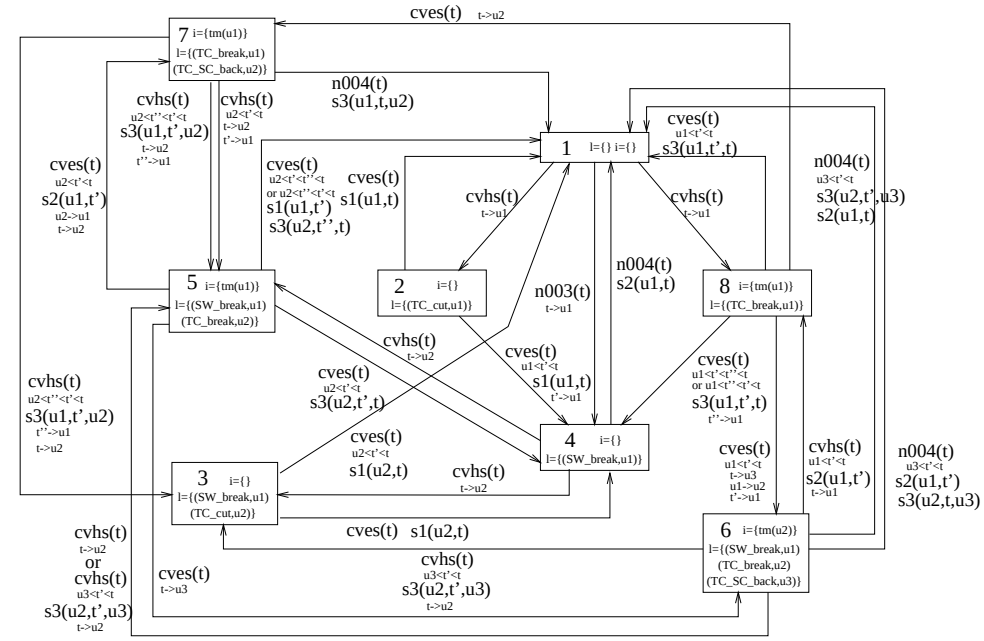
According to the model of the system three alarms can be observed after state 1 : $n003$, $cvhs$ and $cvhs$. The first one means that the system is in state 4 and that a SW_break has occurred. therefore a transition going from state $(1, \{\}, \{\})$ to state $(4, \{SW_break\}, \{\})$ and labeled by $n003$ is added to the diagnoser. As SW_break and $n003$ are on the same model transition, their occurrence time are equal. If $n003$ is received at time $u1$, SW_break occurs at time $u1$: this temporal information is added to the diagnoser.

Let us look at the $cvhs$ alarm that leads from state 1 to state 2 in the model. When this alarm is received, the system goes to state 2 and a TC_cut occurs. Therefore a transition from state $(1, \{\}, \{\})$ to state $(2, \{TC_cut\}, \{\})$ is added to the diagnoser. The date associated to TC_cut is the date of the alarm $cvhs$.

The alarm $cvhs$ leads from state 1 to state 8 in the model. When this alarm is received, the system is in state 8, a TC_break has occurred and an internal event has been emitted. Therefore a transition from state $(1, \{\}, \{\})$ to state $(8, \{SW_break\}, \{TC_SW_back\})$ is added to the diagnoser. If $cvhs$ is received at time $u1$ then SW_break has occurred at time $u1$ and TC_SW_back has to occur between $u1 + d1$ and $u2 + d2$.

All the possible transitions going out of the initial state have been added to the diagnoser. Three new states have been built : $(4, \{SW_break\}, \{\})$, $(2, \{TC_cut\}, \{\})$ and $(8, \{SW_break\}, \{TC_SW_back\})$. The same work has to be done for each of these states and so on. Finally we obtain the diagnoser given by figure 19.

The way the diagnoser is used on-line will be presented at the end of the next section devoted to the generic diagnoser.



Notations :

[d1,d2], [d'1, d'2], [d"1,d"2] denote the time intervals corresponding respectively to the SW rebooting delay, the TC repairing delay, the SW repairing delay.

s1(u1,u2) = ((TC_cut,u1), (TC_SC_back,u2)) with d'1 < u2-u1 < d'2

s2(u1,u2) = ((SW_break,u1), (SW_TC_back,u2)) with d''1 < u2-u1 < d''2

s3(u1,u2,u3) = ((TC_break,u1), (TC_SW_back,u2), (TC_SC_back,u3)) with d'1 < u3 - u1 < d'2 and d1 < u2 - u1 < d2

tm(u) = (TC_SW_back,u+d1,u+d2)

Figure 19: Extended diagnoser of a system composed of a technical center TC and a switch SW

5 A generic diagnoser

5.1 Motivations

In [21], the diagnoser is built from a model of the whole system. Starting from discrete event models of the individual components, a systematic procedure for generating the global model is presented. It is based on the standard synchronous composition operation. The model captures the interaction between the components and represents the whole system to be diagnosed.

In our case the system also consists of individual components. Starting from the automaton of each of these components we can generate a model of the whole network, using the composition operation described in [18, 5]. But even when considering only TCs and SWs, the model of the whole network has $2^{10}4^{300}$ states. It is clearly impossible to use such a model.

To solve this problem, we take advantage of the specific structure of the network, in our case a hierarchical structure. Instead of dealing with the model of the whole network, we make use of the model of a generic component of the system, in our case a branch of the network. This generic model is obtained, as explained in 2.4, by composing the models of the components of the branch. For example, for the network shown in figure 20 which has two technical centers and five switches, the generic model results from the composition of the models of the supervision center, one technical center and one switch. This model is generic as it can be used to model each branch of the network. In this example, the generic model can be used to model $b1, b2, \ldots$ and $b5$. The state of the network is fully described by the state of its branches.

A generic component is a component which appears many times in the network, with the same behavioral model. The use of generic models is then not restricted to hierarchically structured systems but can be extended to other structured systems.

We explain in the following how the state of a branch can be economically represented by using adequate primitives and operators (section 5.2) and then how the generic diagnoser will compute the current states of the network according to received alarms (section 5.4).

5.2 Economical representation of the state of the network

From a generic model, it is easy to build a generic diagnoser as explained in the previous section. Before explaining the use of this generic diagnoser to monitor the whole network, an important point is to focus on the way the states of the branches of the network are economically represented. Since the effects of an alarm depend on the state of the branches from which it has been emitted, we have to store all possible states of the network branches. Dealing with branches does not make the combinatorial problem associated to the number of possible states disappear; for one technical center and thirty switches, more than 4^{30} states are possible. It is then crucial to have an economical and generic representation of the network, relying on a partition of the network's branches. All the branches which share the same set of possible states will be represented only once, being described by one partition.

Moreover, to prevent enumerating the branches, they will be represented on a relational form. For example, *branch_of* associates to a component all the branches passing through the component. In figure 20, $branch_of(TC2)$ is $\{b3, b4, b5\}$. The state of the network can be described by $\{(branch_of(TC1), setofstate1), (branch_of(TC2), setofstate2)\}$ where $setofstate$ describes diagnoser states; it means that $b1$ and $b2$ are in $setofstate1$ and that $b3$, $b4$ and $b5$ are in $setofstate2$[6]. The supervision problem will consist in updating this state on the reception of new alarms.

5.3 Definitions

Some definitions required for describing more formally the use of the generic diagnoser are given in this section. Each definition is illustrated with an example using the network described by figure 20.

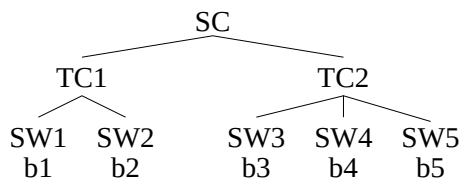


Figure 20: Structure of an illustrative network

Definition : $SET_OF_COMPONENTS$
$SET_OF_COMPONENTS$ is the set of components of the network. In the example, $SET_OF_COMPONENTS$ is $\{SC, TC1, TC2, SW1, SW2, SW3, SW4, SW5\}$.

Definition : son_of
$son_of \in SET_OF_COMPONENTS \rightarrow \mathcal{P}(SET_OF_COMPONENTS)$
son_of is used to describe the hierarchical structure of the network. For the network described by figure 20, son_of is given by:

$son_of(SC) = \{TC1, TC2\}$,
$son_of(TC1) = \{SW1, SW2\}$,
$son_of(TC2) = \{SW3, SW4, SW5\}$,
$\forall i \in [1..5]\ son_of(SWi) = \emptyset$

Definition : branch
A branch of the network is a set $b = \{SC, c_1, c_2, \ldots, c_n\}$ where $c_1 \in son_of(SC)$, $c_i \in son_of(c_{i-1})$ and $son_of(c_n) = \emptyset$. In the example $b1 = \{SC, TC1, SW1\}$ is a branch of the network shown by figure 20.

Definition : Ω
Ω is the set of all the network's branches. In the example Ω is $\{b1, b2, b3, b4, b5\}$.

[6]For understandability, in the examples we will often give enumerated sets. In the prototype sets are never enumerated: the relation *branch_of* is always used

Definition : branch_of
$branch_of \in SET_OF_COMPONENTS \rightarrow \mathcal{P}(\Omega)$
$branch_of$ associates to a component c all the branches that include c.
$branch_of(c) = \{b \in \Omega / c \in b\}$.

The relation $branch_of$ prevents of enumerating all the branches. For example, when all the branches under $TC1$ (in the real network there are 30 branches if we only consider TCs and SWs) are in a same set of states, we do not enumerate the set of branches but we just use $branch_of(TC1)$.

Definition : Set of dated events
A set of dated events is a set of pairs including an event and a time. The set $\{(ev_1, t_1), (ev_2, t_2), \ldots, (ev_n, t_n)\}$ means that the events ev_i occur at time t_i.

Definition : Historical record
An historical record looks like $[h_1, h_2, \ldots, h_n]$ and describes a set of alternative histories; each history h_i is a set of temporally constrained events.

For example, $[\{(ev_1, t_1), (ev_2, t_2)_{\mathcal{C}_{12}}\}, \{(ev_3, t_3)_{\mathcal{C}_3}\}]$ means that ev_1 occurred at time $t1$ and ev_2 at time $t2$ with $t1$ and $t2$ satisfying the constraints $\mathcal{C}_{12}$ or that ev_3 occurred at time t_3 with $t3$ satisfying the constraints $\mathcal{C}_3$.

Definition : State of the network
The state of the network is described by a set $E = \{(P_1, E_1), \ldots, (P_n, E_n)\}$ where P_i is a set of branches and $E_i = \{(e_{i1}, h_{i1}), \ldots, (e_{in}, h_{in})\}$ is a set of pairs (state of the generic diagnoser, historical record). Each branch of P_i is in one of the possible states described by E_i, i.e in one of the state e_{ij} with the history h_{ij}.

For example, let us consider the network given in figure 20, the generic diagnoser of a branch of this network is given by figure 19. This diagnoser has been computed by composing the model of a TC and an SW. As the supervision center has only one state, the model used is also the model of an SC, a TC, and an SW. Let us assume that each component of the system is in a normal state: each branch of the network is in the initial state of the diagnoser (state 1)[7]. Therefore the state of the network is given by $E = \{(branch_of(SC), \{(1, [])\})\}$.

Let us now explain how the state of the network is updated by the generic diagnoser on the occurrence of an alarm.

5.4 Updating the current state of the network

The supervision problem is to update the current state of the network according to the flow of received alarms. In this subsection, the network is assumed to be in state $E = \{(P_1, E_1), \ldots, (P_n, E_n)\}$ with $E_i = \{(e_{i1}, h_{i1}), \ldots, (e_{in}, h_{in})\}$. We present how the new state E' of the network is computed when an alarm a, emitted by a component co, is received.

The fact that the component co emitted an alarm means that it detected a problem referred in the following as a failure. Let $B(a)$ be the set of generic components that may

[7] A diagnoser state is made up of a number, a set of failures and a set of internal events. In the diagnoser of the figure 19, the number is sufficient to identify the state of the diagnoser. Therefore in all the examples dealing with this diagnoser we will only use the number.

be affected by this failure and $\overline{B(a)}$ the set of components which are surely not affected by it. In our application, the only components hierarchically under *co* can have been affected by the failure. Therefore $B(a) = branch_of(co)$ which is the set of all branches including *co*. In the network represented by figure 20, if $TC2$ emits an alarm *cvhs*, $B(cvhs(TC2)) = branch_of(TC2) = \{b3, b4, b5\}$.

When updating the state of the network on the reception of an alarm a, each element P_i of E will be differently considered, according to its belonging to $B(a)$ or to $\overline{B(a)}$.

5.4.1 Elements of $P_i \cap \overline{B(a)}$:

A generic component which belongs to $\overline{B(a)}$, with $\overline{B(a)} = \Omega - B(a)$, is not affected by the failure. Therefore it stays in the same set of possible states E_i. For example, let us suppose that the state of the network shown in figure 20 is $E = \{(branch_of(SC), \{(1, [])\})\}$. When the alarm *cvhs* emitted by $TC2$ is received, the set of possible states of all the branches which do not belong to $branch_of(TC2)$ does not change. It means that the new set of possible states of the branches denoted by $branch_of(TC1)$ are $\{(1, [])\}$.

5.4.2 Elements of $P_i \cap B(a)$:

Let $b \in P_i \cap B(a)$. The component b may have been affected by the failure. Therefore the diagnoser is used to compute the new possible states of b.
Let $E'_i = \bigcup_{j \in [1..n]} \bigcup_{k \in [1..m]} (\delta_{dk}(e_{ij}, a), add(h_{ij}, new_emission_{jk}))$ where $\delta_{dk}(e_{ij}, a)$ denotes one of the m diagnoser states reachable from e_{ij} by a, $new_emission_{jk}$ denotes the set of dated events emitted by this transition and $add(h_{ij}, new_emission_{jk})$ adds to each history of h_{ij} the set $new_emission_{jk}$. The set E'_i corresponds to all the new possible states of b. It can be empty when there is no state reachable from any e_{ij} by a. In the example, the concerned branches are those belonging to $branch_of(SC) \cap branch_of(TC2)$, i.e the branches denoted by $branch_of(TC2)$. The accessible states from 1 by *cvhs*, $\delta_d(1, cvhs)$, are the states 2 and 8. There are no emitted events on these transitions. The new set of possible states for $branch_of(TC2)$ is $\{(2, []), (8, [])\}$.

5.4.3 Computing the new state:

Let the state be $E = \{(P_1, E_1), \ldots, (P_n, E_n)\}$ and the received alarm a. A first step consists in evaluating all $E'_1, E'_2, \ldots, E'_n$ as explained in 5.4.1 and 5.4.2. Let then ST be $\{(P_1 \cap B(a), E'_1), (P_1 \cap \overline{B(a)}, E_1), \ldots, (P_n \cap B(a), E'_n), (P_n \cap \overline{B(a)}, E_n)\}$. In the cases where either $P_i \cap B(a)$ or $P_i \cap \overline{B(a)}$ is empty, the corresponding pairs are deleted from the set ST. In the case where E'_i is empty, the corresponding pairs are also deleted from ST. Lastly, the temporal constraints associated to the events are checked and the inconsistent pairs are deleted.

5.4.4 Extracting histories detected as diagnostic candidates:

The pairs $(state, historicalrecord)$ give for each branch, and then for each component of the branch, the possible sequences of events currently explaining all the observations. It can then be seen as the current diagnostic candidates for the branch. Under some conditions, it can be guaranteed that no future observation will question this current set of diagnostic candidates. Each history can definitively be considered as one possible candidate explaining the alarms received at this date. To get a more economical representation of the network state, these candidates are stored in a dedicated data structure $HIST$ which collects 4-tuple $(P, t, e, [h_1, h_2, \ldots, h_n])$ where P describes a set of branches , t is the current date, e is a diagnoser state and $[h_1, h_2, \ldots, h_n]$ the corresponding historical record. The network state is consequently updated by reinitializing the stored historical records. The conditions under which such a storage operation can be performed are not, in general, easy to determine. The branch must be associated to a unique diagnoser state; this state must be a *stable* state, i.e a state with no internal events and no temporal constraints. In our application, it is the case for the diagnoser state which corresponds to the normal state of the system and contains neither expected internal events nor temporal constraints (diagnoser state 1 in Figure 19).

5.4.5 The join operation:

The next step is to examine ST in order to identify whether two sets of branches can be *joined* into one. The idea is to merge the sets of branches which have the same set of possible states and identical histories. To simplify, we apply the join operation only to the case where the historical records are empty sets.

Let (BR_1, E_1) and (BR_2, E_2) be two elements of ST. BR_1 and BR_2 are sets of branches. $E_1 = \{(e_{11}, []), \ldots, (e_{1n}, [])\}$ and $E_2 = \{(e_{21}, []), \ldots, (e_{2n}[])\}$ are sets of pairs (diagnoser state, empty historical record). BR_1 and BR_2 can be joined if and only if $\forall i \in [1..n] \exists j \in [1..n] e_{1i} = e_{2j}$. We get $(BR_1 \cup BR_2, \{(e_{11}, []), \ldots, (e_{1n}, [])\})$.

5.4.6 Summary

When an alarm a is received, the following algorithm is used to update the state $E = \{(P_1, E_1), \ldots, (P_n, E_n)\}$ into the new state E'.

Compute $B(a)$
For all $i \in [1..n]$ compute E'_i (see 5.4.1 and 5.4.2)
ST=$\{(P_1 \cap B(a), E'_1), (P_1 \cap \overline{B(a)}, E_1), \ldots, (P_n \cap B(a), E'_n), (P_n \cap \overline{B(a)}, E_n)\}$
For all $i \in [1..n]$
  if (emptyset$(P_i \cap B(a))$) then $ST = ST - (P_i \cap B(a), E'_i)$
  if (emptyset$(P_i \cap \overline{B(a)})$ then $ST = ST - (P_i \cap \overline{B(a)}, E_i)$
  Update E'_i by deleting the temporally inconsistent states
  if (emptyset(E'_i)) then $ST = ST - (P_i \cap B(a), E'_i)$
Apply the store operation with update of $HIST$ and ST (see 5.4.4)
Apply the join operation on ST (see 5.4.5)

$E' := ST$

5.5 Example

We consider the network given by figure 20 and the extended diagnoser of figure 19 and show how the network state evolves when the following sequence of alarms is received: $(cvhs(TC2), 10)$, $(cves(TC2), 20)$ $(n004(SW4), 25)$, $(n003(SW1), 140)$. The delay d_{max} corresponds to the maximal delay for any component before it is repaired after a breakdown ($d_{max} = d'2 = d''2$; $d'1 = d'2 = 0$). This delay is much greater than the maximal delay associated to internal events. In the following example, we suppose that $d_{max} = 100$ and that the delay taken by the switch number 4 to reboot belongs to [5..10]. These temporal constraints are not explicitly stated in the following example but are checked according to the above algorithm.

For sake of simplification, we use the shortened notation of figure 19.

- The initial state of the network is $E = \{(\Omega, \{(1, [])\})\}$ and $HIST = \emptyset$.
- Reception of the alarm $a = (cvhs(TC2), 10)$
  We have $B(a) = branch_of(TC2)$.
  - $\Omega \cap \overline{B(a)} = branch_of(TC1)$
    All the branches of $branch_of(TC1)$ keep the same set of possible states and histories. We get then $\{(1, [])\}$.
  - $\Omega \cap B(a) = branch_of(TC2)$
    The new candidates for the branches of $branch_of(TC2)$ correspond to the diagnoser states 2 or 8 with empty historical records. We get $\{(2, []), (8, [])\}$
  - No storage operation; no join operation.

  The updated state of the network at time 10 is then:
  $E' = \{(branch_of(TC1), \{(1, [])\}), (branch_of(TC2), \{(2, []), (8, [])\})\}$
- Reception of the alarm $a = (cves(TC2), 20)$
  We have $B(a) = branch_of(TC2)$.
  - $branch_of(TC1) \cap \overline{B(a)} = branch_of(TC1)$
    All the branches of $branch_of(TC1)$ keep the same set of possible states and histories. We get then $\{(1, [])\}$.
  - $branch_of(TC1) \cap B(a) = \emptyset$
  - $branch_of(TC2) \cap \overline{B(a)} = \emptyset$.
  - $branch_of(TC2) \cap B(a) = branch_of(TC2)$
    The new candidates for these branches are:
    $\{(1, [\{s1(10, 20)\}]), (4, [\{s1(10, 20)\}]), (4, [\{s3(10, t, 20)_{(10<t<20)}\}]),$

$(6,[]),(7,[]),(1,[\{s3(10,t,20)_{(10<t<20)}\}])\}$
$= \{(1,[\{s1(10,20)\},\{s3(10,t,20)_{(10<t<20)}\}]),$
$(4,[\{s1(10,20)\},\{s3(10,t,20)_{(10<t<20)}\}]),(6,[]),(7,[])\}$

– No storage operation; no join operation.

The new updated state of the network at time 20 is then:
$E' = \{(branch_of(TC1),\{(1,[])\}),$
$(branch_of(TC2),\{(1,[\{s1(10,20)\},\{s3(10,t,20)_{(10<t<20)}\}]),$
$(4,[\{s1(10,20)\},\{s3(10,t,20)_{(10<t<20)}\}]),(6,[]),(7,[])\})\}$

- Reception of the alarm $a = (n004(SW4),25)$
  We have $B(a) = branch_of(SW4)$.
  - $branch_of(TC1) \cap \overline{B(a)} = branch_of(TC1)$
    All the branches of $branch_of(TC1)$ keep the same set of possible states and histories. We get then $\{(1,[])\}$.
  - $branch_of(TC1) \cap B(a) = \emptyset$
  - $branch_of(TC2) \cap \overline{B(a)} = branch_of(SW3) \cup branch_of(SW5)$. All these branches keep the same set of possible states with the same historical records. We get then:
    $\{(1,[\{s1(10,20)\},\{s3(10,t,20)_{(10<t<20)}\}]),$
    $(4,[\{s1(10,20)\},\{s3(10,t,20)_{(10<t<20)}\}]),(6,[]),(7,[])\}$
  - $branch_of(TC2) \cap B(a) = branch_of(SW4)$
    For these branches the new diagnostic candidates are: $\{(1,H)\}$ where
    $H = [\{s1(10,20),s2(t,25)_{(10<t<20)}\},$
    $\{s2(t,t')_{(10<t<20<t'<25)},s3(10,25,20)\},$
    $\{s3(10,t,20)_{(10<t<20)},s2(t',25)_{(t<t'<20\vee 10<t'<t)}\},$
    $\{s3(10,t',20)_{(20<t'<25)},s2(t,25)_{(10<t<20)}\},$
    $\{s3(10,25,20)\}]$
  - The diagnostic candidates $\{(1,H)\}$ of the branches $branch_of(SW4)$ can be stored in $HIST$: $HIST \leftarrow HIST \cup \{(branch_of(SW4),25,1,H)\}$. The historical records are reinitialized and the new candidates are then $\{(1,[])\}$.
  - $branch_of(TC1)$ and $branch_of(SW4)$ can be joined as their candidates satisfy the conditions: their diagnoser states are the same and the historical records are empty. We get:
    $\{(branch_of(TC1),\{(1,[])\}),(branch_of(SW4),\{(1,[])\})\} \xrightarrow{\text{join}}$
    $\{(branch_of(TC1) \cup branch_of(SW4),\{(1,[])\})\}$

The updated state of the network at time 25 is then:
$E' = \{(branch_of(TC1) \cup branch_of(SW4),\{(1,[])\}),$
$(branch_of(SW3) \cup branch_of(SW5),$
$\{(1,[\{s1(10,20)\},\{s3(10,t,20)_{(10<t<20)}\}]),$
$(4,[\{s1(10,20)\},\{s3(10,t,20)_{(10<t<20)}\}]),(6,[]),(7,[])\}$

- Reception of the alarm $a = (n003(SW1), 140)$
  We have $B(a) = branch_of(SW1)$.
  - $(branch_of(TC1) \cup branch_of(SW4)) \cap \overline{B(a)} =$
    $branch_of(SW2) \cup branch_of(SW4)$
    Their new candidates are $\{(1, [])\}$.
  - $(branch_of(TC1) \cup branch_of(SW4)) \cap B(a) = branch_of(SW1)$
    Their new candidates are $\{(4, [])\}$.
  - $(branch_of(SW3) \cup branch_of(SW5)) \cap \overline{B(a)} = branch_of(SW3) \cup$
    $branch_of(SW5)$. The set of their past possible states is:
    $\{(1, [\{s1(10, 20)\}, \{s3(10, t, 20)_{(10<t<20)}\}]), (4, [\{(s1, 10, 20)\},$
    $\{s3(10, t, 20)_{(10<t<20)}\}]), (6, []), (7, [])\})\}$.
    Some of these states are no longer consistent with the delay constraints and can therefore be eliminated. States 6 and 7 are ruled out on account of the internal delay: the internal events, TC_SW_back, corresponding to the end of $SW3$ and $SW5$ rebootings were expected to occur before 40. State 4 is ruled out on account of the maximal delay d_{max}: the repair events were expected to occur before 120. Therefore the new diagnostic candidates of this set of branches are $\{(1, [\{s1(10, 20)\}, \{s3(10, t, 20)_{(10<t<20)}\}])\}$.
  - $(branch_of(SW3) \cup branch_of(SW5)) \cap B(a) = \emptyset$
  - The diagnostic candidates of $branch_of(SW3) \cup branch_of(SW5)$ can be stored in $HIST$:
    $HIST \leftarrow HIST \cup \{(branch_of(SW3) \cup branch_of(SW5), 140, 1,$
    $[\{s1(10, 20)\}, \{s3(10, t, 20)_{(10<t<20)}\}])\}$.
    Their historical records are reinitialized and we get $\{(1, [])\}$.
  - $branch_of(SW2) \cup branch_of(SW4)$ and $branch_of(SW3) \cup branch_of(SW5)$ can be joined for their diagnostic candidates satisfy the conditions:
    $\{(branch_of(SW2) \cup branch_of(SW4), \{(1, [])\}),$
    $(branch_of(SW3) \cup branch_of(SW5)\{(1, [])\})\} \xrightarrow{\text{join}}$
    $\{(branch_of(SC) - branch_of(SW1), \{(1, [])\})\}$

  The updated state of the network at time 140 is then:
  $E' = \{(branch_of(SC) - branch_of(SW1), \{(1, [])\}), (branch_of(SW1), \{(4, [])\})\}$.

6 Implementation and results

6.1 Implementation

The implementation task was carried out step by step. In a first step, the modelling part was implemented. The model itself was tuned up by using a simulation tool and validated on a test file, as explained in 6.2. In a second step, the *Dyp* prototype was developed. *Dyp*

implements the extended diagnoser approach without taking into account temporal delays. From the description of the structure of the network and from the model of each component, it builds the global model of the network and its global diagnoser. The diagnoser state can be tracked as it analyses a stream of alarms. The diagnostic candidates are stored in an historical file and can be displayed. Figure 21 shows the main window of *Dyp*. In a third step, the generic diagnoser, *Gen*, has been implemented. *Gen* is based on *Dyp* but relies on a generic model of the network. It allows to track the state of the whole network as alarms are received. The diagnostic candidates (set of branches, historical record) are stored in a file and can be displayed. It was also experimented on the test file. We are currently implementing the temporal version of the generic diagnoser.

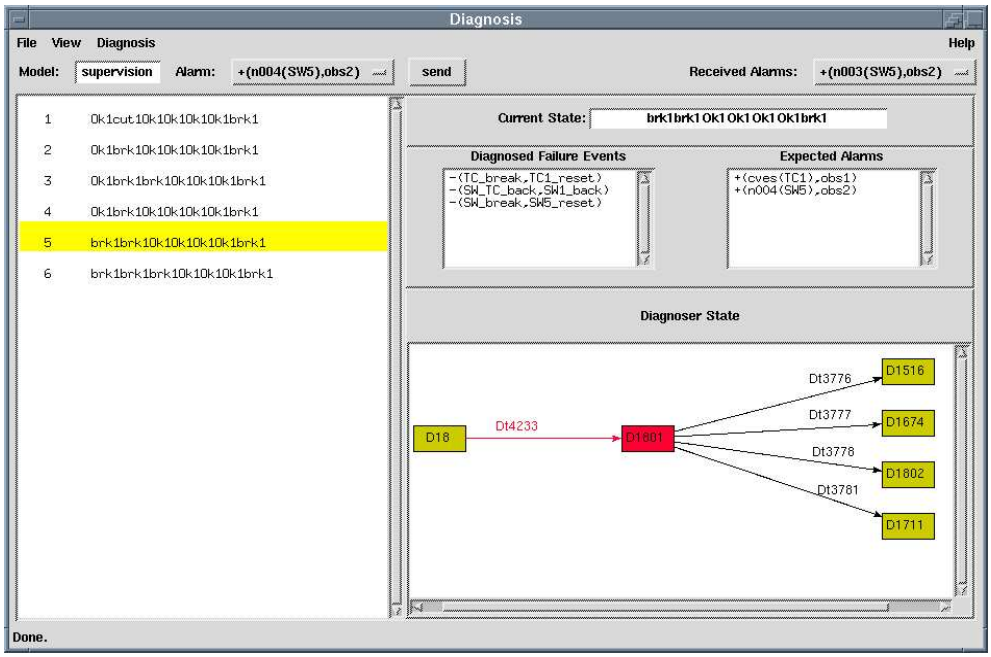


Figure 21: Main window of *Dyp*

In parallel, we are currently experimenting another way of dealing with the large size of the model by designing a decentralized diagnoser approach. The general idea is to rely on local diagnosers built from component models. A coordinator is in charge of computing, step by step, the global diagnoses from these local diagnoses. This decentralized approach is described in [16].

Figure 22: Windows of *Gen*

6.2 Test and comparison with the expert system

The work presented above has been experimented on real data. A test file including 2052 alarms emitted by a subpart of the network during 12 hours has been used as a reference for validating our prototypes. The file was first analysed by the expert system which identified the following failures: 1 TC_break, 2 SW_break, 3 USG_shift, 5 USG_reconfiguration, 2 USG_break, 1 UE_shift and 2 UE_rescue_break. The same stream of alarms was then analyzed by the generic diagnoser and the results were compared. All the faults detected by the expert system were also proposed as diagnostic candidates by the diagnoser which is quite satisfactory. However, in absence of temporal constraints, one problem is the size of the historical records associated to the candidates. It is clear that temporal constraints are a good mean of pruning a lot of candidates which are no longer temporally valid. The implementation and the experimentation of the temporal generic diagnoser is in progress.

The intrinsic quality of the results is difficult to access because it requires to find experts (at least one) accepting to study the alarm files and to qualify the results with respect to their own analysis of the situation. It is then difficult to compare these two approaches more precisely on their results. A clear advantage of our model-based approach on the expert system one is the possibility of taking into account behavioral and topological changes in the network. As it is well-known, any change in the network components or any change in the connections between them makes the expert system obsolete : the expertise has to be updated and in most of the cases reacquired from scratch. These knowledge-based

approaches, attractive in stable domains as medecine, are not adequate for quickly evolving domains as telecommunication networks. In model-based approaches, the changes are made on the elementary structural or behavioral models which is in general much easier and the diagnoser is automatically updated.

7 Related work

Detection and isolation of failures in large and complex systems such as telecommunication networks are crucial and challenging tasks. The generality, flexibility and reliability of model-based approaches make them a very desirable way to monitor these complex systems. In [20] an interesting approach has been proposed, known as the diagnoser approach. As explained above (see section 3.2), this approach relies on a discrete-event model of the system to be surveyed and transforms it, off-line, into a finite state machine that is suitable for analyzing observables and recognizing the faults. Its initial application was a heating, ventilation and air-conditioning unit.

[9] extends this approach to systems whose temporal behavior is of great concern. They propose to combine the timed discrete-event system framework described by [7] with the automata-based diagnoser of [20]. This extension to temporal information is interesting but was applied to a relatively simple problem, the factory conveyor example. They do not provide any solution to the problem raised by the huge size of the global model as soon as real problems are tackled.

[10] investigates the problem of diagnosing discrete-event systems under decentralized information. They propose a coordinated decentralized architecture consisting of local sites communicating with a coordinator responsible for the final diagnosis. A protocol is defined which specifies the diagnostic information generated at each local site, the communication rules used by the local sites and the decision rule employed by the coordinator. Each on-line diagnostic process is carried out by diagnosers similar to those of [20]. The same restrictive assumptions are made: no cycles of unobservable events, and perfect reliability of the communications between the sites and the coordinator (no loss of alarms and arrival in the same order as they were emitted). Moreover, each site is assumed to have at its disposal the global model of the system which, in practice, is nearly always impossible due to the size of this model.

[2] propose a model-based approach for large distributed discrete-event systems (they call them active systems) and rely, as we do, on communicating automata to model the behavior of the components. Starting from the remark that the use of a global model is almost invariably impossible in practice, the main advantage of their proposal is that they do not need any global diagnoser to be built. Their approach relies on the on-line and incremental reconstruction of the behavior of the system, guided by available observations. It consists therefore in generating all the possible histories that are consistent with a given set of observations. The main problems remains the possibly huge size of the search space, but focusing techniques can be used to limit it.

The combinatorial complexity associated with model-based techniques is one of the main problem when dealing with complex systems. It is why it is often more realistic to use the model off-line in order to produce some task-oriented data structure (in our case a diagnosis-oriented structure) which will be used on-line. It is the main idea supporting the diagnoser approaches. Another way of using the model off-line is by acquiring expertise rules from the model by simulating faults. For example in [8, 15], a fault diagnosis knowledge base is built from a qualitative model of the device. This is achieved by systematically simulating all component failures and by learning production rules using an IDL algorithm. A similar approach can be used to acquire scenarios[8], as was done in [4]. In this prototype called GASPAR, we investigated a twofold architecture. The off-line part was in charge of building a scenario base by simulating faults and then learning discriminating scenario. The on-line part uses a scenario recognition system in order to recognize failures on the basis of the stream of alarms. The GASPAR system was tested on the same application as the one presented in this paper, i.e supervising telecommunication networks. A similar approach described in [13] was applied to power distribution networks.

8 Conclusion

This paper proposes a model-based approach for the monitoring of large systems such as telecommunication networks. Our work is based on the diagnoser approach proposed by [20]. Starting from a discrete-event model of the network adapted to simulating faults, this approach consists in transforming the model into a finite state automaton, called a diagnoser, adapted to analyzing on-line the stream of alarms. After presenting our application, we outlined the main limitations of the initial proposal with respect to the difficulties we had to cope with and explained how we overcame them by extending it.

The first contribution of this paper is to propose an extension of the diagnoser approach which is able i) to deal with *communicating* finite state machines, ii) to take into account temporal information, iii) to overcome the case of masked alarms. A definition of an extended diagnoser and a schematic algorithm to build it are given. The second contribution is to show that relying on a global model of the system is in general impossible when concerned with complex systems such as telecommunication networks. We have proposed a solution which takes advantage of the (in our case hierarchical) structure of the system and relies on a *generic* model of the system. An adequate representation based on manipulating formal sets provides an economical representation of the current state of the network. A *generic diagnoser* able to deal with such a generic model of the system is then proposed.

Our approach has been implemented and experimented on the application, i.e. the largest French packet switching telecommunication network. A current work consists in adding preferences in order to prune the candidates and to more efficiently track the system. In parallel, we are currently experimenting another way of dealing with the large size of the

[8] A scenario (or chronicle) as defined for example by [12] is a set of events associated to a set of temporal constraints. Given a set of scenarios (characteristic of faults), a scenario recognition system analyze on-line sequence of events and identify corresponding fault situations.

model by designing a decentralized diagnoser approach. The general idea is to rely on local diagnosers built from component models. A coordinator is in charge of computing, step by step, the global diagnoses from these local diagnoses. This decentralized approach is described in [16].

References

[1] R. Alur and D.L. Dill. Automata for modeling real-time systems. In *Proceedings of ICALP'90, LNCS 443*, pages 321–335. Springer-Verlag, 1990.

[2] P. Baroni, G. Lamperti, P. Pogliano, and M. Zanella. Diagnosis of active systems. *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 274–278, 1998.

[3] S. Bibas. *Simulation à base de modèles pour la supervision de réseaux de télécommunication.* Phd thesis, Université Paris-Nord, 1997.

[4] S. Bibas, M.O. Cordier, P. Dague, F. Lévy, and L. Rozé. Gaspar: a model-based system for diagnosing telecommunication networks. In *IMACS-IEEE/SMC International Multiconference of Computational Enfineering in Systems Applications (CESA'96)*, pages 338–343, Lille, France, 1996.

[5] S. Bibas, M.O. Cordier, P. Dague, F. Lévy, and L. Rozé. Modelling a telecommunication network for supervision purposes. In *Proceedings of the ECAI96 Workshop on Model-Based Systems and Qualitative Reasoning - Perspectives for Industrial Applications*, pages 160–166, Brighton, UK, 1996.

[6] D. Brand and P. Zafiropulo. On communicating finite state machines. *Journal of ACM*, 30:323–342, 1983.

[7] B.A. Brandin and W.M. Wonham. Supervisory control of timed discrete event processes. *Proceedings of the IEEE Transactions on Automatic Control*, 39(2):329–342, 1994.

[8] I. Bratko, I. Mozetic, and N. Lavrac. Automatic synthesis and compression of cardiological knowledge. In *Machine Intelligence*, volume 11, pages 435–454. Ellis Horwood, 1988.

[9] Y.L. Chen and G. Provan. Modeling and diagnosis of timed discrete event systems - a factory automation example. *Proceedings of the American control conference*, 1997.

[10] R. Debouk, S. Lafortune, and D. Teneketzis. A coordinated decentralized protocol for failure diagnosis of discrete-event systems. In *4th International Workshop on Discrete-Event Systems*, pages 138–143, Cagliari, Italy, 1998.

[11] R. Debouk, S. Lafortune, and D. Teneketzis. Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Discrete Event Dynamic Systems: Theory and Applications*, 10:33–79, 2000.

[12] 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, 1993.

[13] P. Laborie. Automatic generation of chronicles and its application to alarm processing in power distribution systems. In *Proceedings of the International Workshop on Principles of Diagnosis(DX'97)*, Mont St Michel, France, 1997.

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

[15] D.A. Pearce. The induction of fault diagnosis systems from qualitative models. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 353–357, St Paul, 1988.

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

[17] 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.

[18] L. Rozé. *Supervision de réseaux de télécommunications : une approche à base de modèles.* Phd thesis, Université de Rennes 1, 1997.

[19] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. A discrete event systems approach to failure diagnosis. In *Proceedings of the International Workshop on Principles of Diagnosis(DX'94)*, pages 269–277, New Paltz,USA, 1994.

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

[21] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Failure diagnosis using discrete event models. *Proceedings of the IEEE Transactions on Automatic Control*, 4(2):105–124, 1996.

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399