Fault Detection in Telecommunication Networks Based on a Petri Net Representation of Alarm Propagation *

Renée Boubour(1), Claude Jard(2)

(1) IRISA/INRIA, Campus de Beaulieu, F-35042 Rennes Cedex, France
(2) IRISA/CNRS, Campus de Beaulieu, F-35042 Rennes Cedex, France

Abstract

This paper presents a new use of safe Petri nets and its partial order semantics, in the field of telecommunication network management. Petri nets are used to provide both a model and an algorithm in fault management domain. First, a model of fault and alarm propagation, based on 1-safe Petri nets, is defined. Then an algorithm, based on net unfolding, is designed. This study leads to a generic supervisor, which can be easily distributed on a set of sensors.

1 Introduction

The complexity of communication networks and the volume of information provided by these networks have caused an increase in demand for network management systems. In particular, the area of network fault management requires a lot of expertise and is known to be difficult to handle. Most of the current proposals are build on an ad-hoc basis. They are also much more involved in structuring the management system than in designing dedicated algorithms. There is a real pressing need for establishing a theoretical foundation of network fault management.

Our paper aims at contributing to this foundation in focusing on the treatment of causal dependencies between alarms and faults. The main idea is to take into account the essential distributed nature of the problem [3]. This will be done by the use of Petri nets and its "true concurrency" semantics. We think it constitutes an original way to attack the usual question of alarm correlation.

Petri nets are well known as powerful and useful model for concurrent systems. We decided to found our approach on the explicit description of fault and alarm propagation using 1-safe Petri nets. It allows us to express and to deal with multiple faults, alarms interleaving,

*This work is supported by France Télécom/CNET, contract 95 1B 151.

and causal dependencies between faults and alarms. The fault detection algorithm, intended to be implemented in a supervisor, is based on an incremental unfolding of the Petri net.

Our investigation is supported by applying this approach to a specific network system: the SDH network (Synchronous Data Hierarchy), in collaboration with CNET (the research center of France-Telecom). The model and algorithm presented here serve also as a basis for a probabilistic approach to fault detection. This study is conducted in an other team of our laboratory [1].

The technical contribution can be announced in three points:

1. a new way of modeling alarm correlation by focusing on causal dependencies between faults and alarms,

2. a dedicated detection algorithm based on an on-line unfolding of Petri nets,

3. making appear good properties of the model and algorithm which prepare the distribution of the algorithm on network sensors.

The rest of the paper is divided into four parts. Section 2 presents the context and hypothesis of the study, based on the real SDH example. Section 3 details the propagation model, its syntax and semantics, which are illustrated. Section 4 presents the main principles of the detection algorithm. We end by discussing some related works in Section 5, before conclusion.

2 Application Example

2.1 SDH Network

SDH (Synchronous Data Hierarchy) was developed to manage optical interconnections as well as existing plesiochronous signals (see e.g. [10]). Advantages of this system are direct access to lower rate channel without demultiplexing the entire signal, enhanced maintenance facilities, simplified evolution to higher interfaces rates and the ability to carry new broadband channels.

This hierarchy is constructed on a basic unit frame, named STM-1 (Synchronous Transport Module level 1). In STM-1 frame, some bytes are allocated to overhead functions (regeneration section overhead, multiplexing section overhead (MS)), others are allocated to administrative units pointers, and the rest of the available capacity carries the payload in administrative units (AU). Higher interface rates, STM-n frames, are formed by byte interleaving n STM-1 frames (n = 4, 16 or 64). STM-n frames have the same structure as STM-1 frames. The STM-n overhead sections are constructed by the multiplexer, independently of the data in STM-1 frames.

The data transported are first encapsulated in containers of fixed size. Every container is also associated to a path overhead, dedicated to network management. The container and its overhead give the virtual container. The start of a virtual container in a frame is indicated by an administrative unit pointer. There are three types of administrative units defined. These allow different sizes of virtual containers, which are multiplexed into STM-n frames.

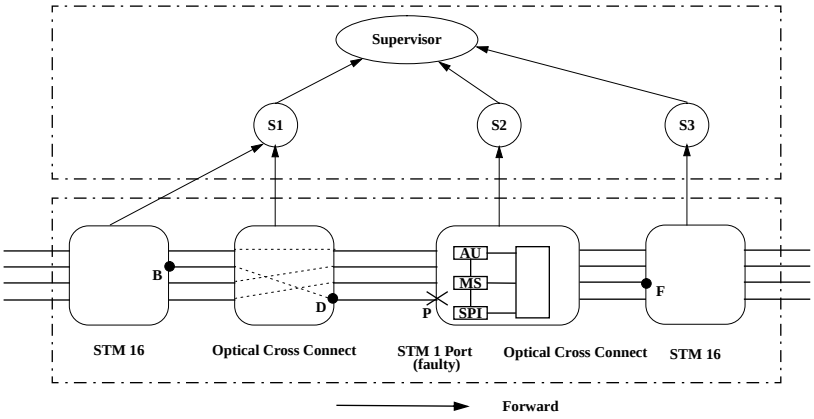

Figure 1: Example of SDH data network and its associated management network

The section overheads and the path overhead allow management of the SDH network at different levels. The multiplexing section overhead is used to manage sections of network between two multiplexers or optical cross connect. The path overhead carries end to end management information about the corresponding container. The path overhead is managed by terminal equipment (as illustrated by STM-16 elements in Fig. 1), which are high level multiplexers.

Figure 1 illustrates a part of an SDH network, with an associated management network. The data network is made of some network elements (STM-16, optical cross connect). These elements are connected via bi-directional connections. Each of these elements contains STM-1 ports. For such a port, there are several components of different levels from management point of view (synchronous physical interface level (SPI), multiplexage section level (MS), administrative unit level (AU)).

A telecommunication network is so viewed as a set of network elements. These elements are grouped into sites, each of them being associated with a sensor. Alarms go from network elements through the sensors s_i to the supervisor (S). In this example, three sites are associated to sensors (s_1, s_2, s_3).

CNET lead a study to highlight propagation of management information in SDH equipments. This study is based on official standards, mainly G774, G782 and G783 from ITU. This study also took advantage of knowledge about real SDH equipments. Its concentrates on fault management, in case of unprotected and bi-directional connections. Standards describe possible faults, and alarm notifications emitted.

When a fault occurs, several alarm notifications are send by the element which detects the fault. Other elements, not directly concerned by the considered fault, also emit notification, because of propagation mechanism. This propagation mechanism uses alarm indication signal (AIS). This signal is obtained by putting all bytes of concerned overhead to value 1. Hence, there are two orthogonal ways of propagation. Firstly, between components of a single network element (such as a port of optical cross connect), AIS is read through components from lower level to higher level. Secondly, AIS goes through link connections and through

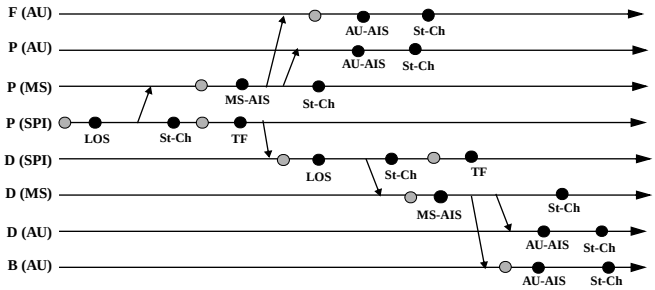

Figure 2: Example of production of alarms and some faults states

cross connections, and is read by higher levels.

2.2 Modeling Fault and Alarm Propagations

In this study, modeling consists in describing expected manifestations from part of the supervised network, like the "event model" described in [14]. It does not consists in describing the whole behavior of the considered system. It appears that, in case of telecommunication networks, dealing with observed manifestations does not lead to much more approximation than abstracting behavior of each component and mix these representations. For example, one should utilize finite state machine to represent, thus abstract, behavior of components. Then a composition is chosen, followed by minimization, to reduce the size of the global machine. This implies a choice for an equivalence. Here, the abstraction is done by the way of choosing significants alarms and fault states.

As a matter of fact, knowledge about faults in telecommunication networks is more often concerned with manifestations caused by faults than with all network elements specifications. For example, standards concerning SDH describe a lot, and maybe too much notifications to send in case of malfunctions. They are not always exhaustively implemented by constructors. Furthermore, because of the size and complexity of networks, the knowledge of an operator is more concerned with the daily received alarms than with the complete set of interactions between all network elements.

The horizontal space-time diagram in Fig. 2 shows alarms (black dots), that are expected in case of a loss of signal (LOS) of the STM-1 port P. Some other elements of the net are involved, which are marked as D (for Distant port), F (for Forward port) and B (for Backward port) in Fig. 1. An horizontal line is dedicated to each involved component. Alarm indication signals are illustrated by oblique arrows in the diagram.

The dashed dots correspond to local states, which could be interesting for an operator to understand what happened, which are fault states to be diagnosed.

In the general case, *alarms* are viewed as special events, signaling faults, by the way of notifications. Alarms propagate, level by level of software components. In Fig. 2, for example, SPI, MS and AU components of port P are involved. Alarm indication signals

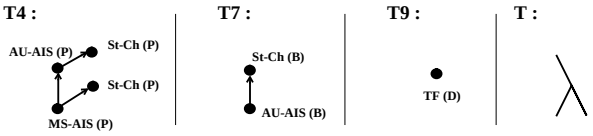

Figure 3: Some examples of alarm patterns

go from low level components (SPI) to higher level components (MS, AU). Alarms also propagate, from a first defect, through physical connections. This is illustrated in the case of LOS with the AIS mechanism. This implies dependencies between alarms. And a fault generally manifest itself through several alarms, with dependency or concurrency between these alarms.

Faults states are viewed as particular local states between alarms. They will constitute the elements of diagnosis. Physically speaking, a fault can occur only once at a time. It makes no sense to consider the same fault involving the same network elements twice at the same time (although it can occur repeatedly). It is pointed out that in telecommunication networks faults could appear simultaneously, i.e. multiple faults occur, which are then concurrent (see e.g. [18]). Moreover, several faults could combine themselves to produce others faults (And, Or dependencies in Fig. 4), and one single fault could result in several other faults (Simultaneous, Exclusive dependencies in Fig. 4).

An alarm pattern could then be defined by a set of alarms (and their dependencies) between two fault states.

2.3 Using Petri Nets and Partial Orders

The previous example allows us to see what it is to be expressed.

It appears that partial orders are adequate to represent dependencies between alarms, as well as concurrencies. Figure 3 illustrates possible dependencies between alarms, coming from the SDH example. λ means that a fault state may not manifest itself, this is the case when fault propagates without notification. Such a partial order on alarms is called an *alarm pattern*, that represents their dependencies. Alarm patterns are associated to fault state manifestation (and so, to fault state propagation).

There are several kinds of dependencies between faults. Places, transitions, and their relationship, appeared to be an adequate tool to express causal dependencies between faults, and multiplicity, as shown in Fig. 4. This figure illustrates elementary cases of causal dependencies between faults, using usual drawing of places and transitions of Petri nets. Persistent faults, spontaneous faults occurrence and reabsorption are also mentioned. Some of them are illustrated with SDH.

A set of dependencies between fault states is therefore represented by a bounded net of capacity one. A fault manifests itself through one, or usually many alarms. So, alarm patterns are associated with transitions of the net. Capacity one is required on places, because of the nature of faults.

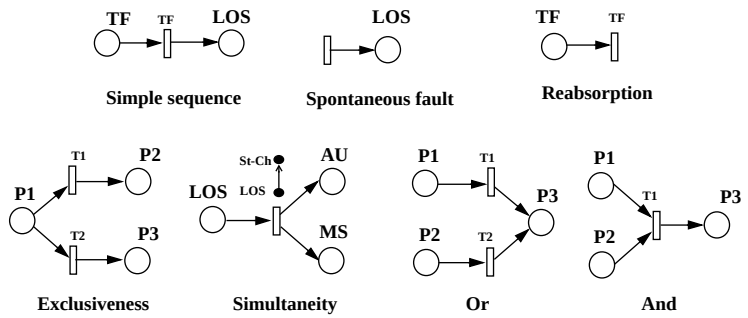

Figure 4: Different kinds of dependencies between faults states in telecommunication networks

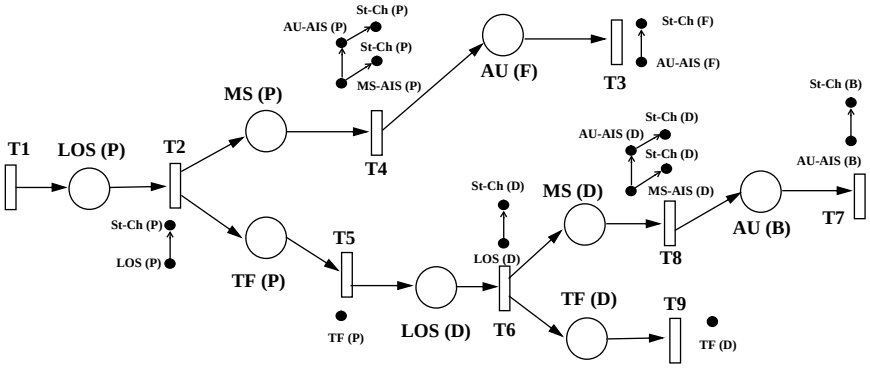

Figure 5: Example of fault net for a LOS (Loss Of Signal) of a STM-1 port

All these considerations lead to finite Petri net, of capacity one, with arc weight 1. Transitions are labeled with alarm patterns when needed. The resulting fault net for the LOS of P is given in Fig. 5.

The so used Petri nets are here called *fault nets*. A 1-safe Petri net can be associated to each fault net. This allows us to use the associated theory and the interesting properties of these nets. All considered nets are finite.

An observation of produced alarms will be provided by the way of a partial order on these alarms, as suggested in Fig. 2. The detection of fault will then consists in recognizing alarm patterns in a given observation. Diagnosis will be provided by a backward simulation of the fault net. The use of probabilities on transitions will allow to deal with approximation due to lack of knowledge or due to inexact observation.

3 Propagation Model

This section gives the formal description of the propagation model. It details the syntax of alarm pattern and fault net. The semantics of the model is then defined in terms of family of finite partial orders on alarms.

3.1 Syntax

An alarm pattern is defined to describe a set of alarms and the expected dependencies between them. An alarm pattern A_i associated to a transition T_i is defined as $(X_i, \leq_i, \mathcal{A}_i, \varphi_i)$, where:

1. X_i as finite set of vertices, i.e. alarms,
2. $\leq_i$ as partial order relation on vertices (where $\leq_i \subseteq X_i \times X_i$), i.e. dependencies between alarms,
3. and φ_i as a labeling function of alarms on $\mathcal{A}_i$.

A vertex represents an occurrence of its labeling alarm. Edges indicate expected dependencies between alarms, according to the knowledge about their possible propagation. In a given alarm pattern, the label $\varphi_i(x) = a$ of an alarm must contain also the sensor $s(x)$ which should observe it. The set of all alarm patterns is $A = \biguplus_i A_i$, where i enumerates transitions of the net. A is the alphabet of alarm patterns.

The covering relation, $\prec\!\!\!-$, on a partial order $O = (X, \leq)$ is defined by $a \prec\!\!\!- b$ if, for all $z \in X$, $a \leq z \leq b$ implies $z = a$ or $z = b$. The set of minimal of O is $Min(O)$, and the set of its maximal is $Max(O)$.

O' is a sub-order of O if $X' \subseteq X$ and $\leq$ in O' is the restriction on X' of $\leq$ in O.

A fault net is defined to describe a set of faults and the known dependencies between them. In a fault net, a place represents a fault state, and transitions represent dependencies between faults. Transitions are labeled by the corresponding alarm patterns, when some manifestations are expected due to this fault state.

Let A be an alarm patterns alphabet. A fault net on A is a 4-tuple $N_F = (P, T, F, l)$ where:

1. P is the finite set of places,
2. T is the finite set of transitions,
3. $P \cap T = \emptyset$,
4. $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation,
5. $l : T \to A \cup \{\lambda\}$ is the labeling function.

3.2 Semantics

3.2.1 Choice of Partial Order Semantics.

As pointed out by Vogler in [22], one needs partial order semantics when considering non-atomic events on transitions. In some sense, this is the case for fault nets, because of alarm patterns on transitions. In case of multiple faults (i.e. several places are active), one must consider concurrency between the associated alarm patterns. The idea is then to work

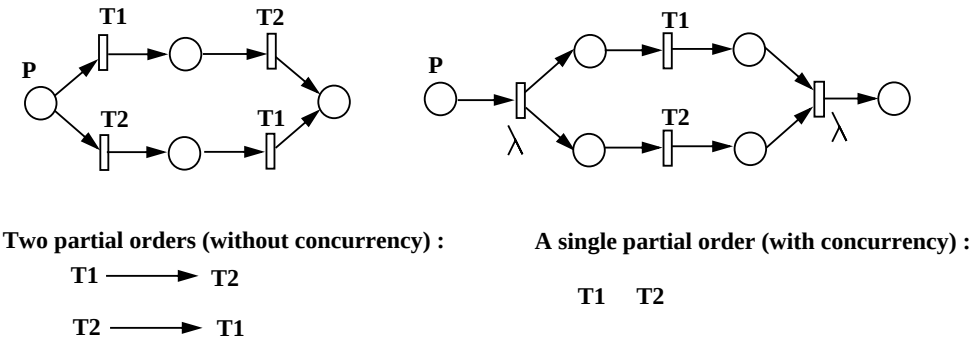

Figure 6: Illustration of partial order semantics

explicitly with the net structure through unfoldings. This will avoid the usual state explosion due to concurrency.

Figure 6 shows two nets, that are indistinguishable with respect to their firing sequences. This is not always true if concurrent firing of transitions is allowed. These nets are different with respect to the set of partial orders on transitions they admit. The net on the left hand admits two partial orders, that are totally ordered, i.e. without concurrency. On the other hand, the second net admits one single partial order, that contains concurrency.

The fault net semantics is defined by the Set of partial Orders on Alarms it describes. Let $\mathcal{S}_{oa}$ denote this semantics. Branching processes of Petri nets and associated event structures are used to represent this semantics of fault net.

3.2.2 From Fault Net to 1-Safe System Net.

For the sake of convenience, a fault net is translated in a *1-safe system net*. This is done in two steps. First, alarm patterns on transitions are expanded onto capacity one nets, in a hierarchical manner. Secondly, places are complemented to code explicitly the capacity one.

As usual, $\forall\ x \in P \cup T$, let ${}^\bullet x = \{y \in P \cup T \mid (y,x) \in F\}$ be its pre-set and $x^\bullet = \{y \in P \cup T \mid (x,y) \in F\}$ its post-set. The marking of a place $p \in P$ is $M(p)$, with values 0 or 1.

To strictly limit the model to 1-safe Petri nets, alarm patterns are expanded onto causal nets. Let A be an alarm pattern. The corresponding labeled net $N(A) = (B, E, F, l)$ is obtained using a construction similar to the construction proposed in [16]:

1. Firstly, create a transition for each vertex of the alarm pattern: for every $a \in X$, consider a unique transition $t(a) = t \in E$. These transitions are labeled with the corresponding alarms : $l(t) = \varphi_i(a)$.

2. Secondly, put a place for every covering edge of the partial order relation. For every $x \prec\!\!\!\cdot\ y$ create a unique $p \in B$ such that ${}^\bullet p = t(x)$ and $p^\bullet = t(y)$.

3. Finally, put a place under every minimal of the order, and similarly, on top of every maximal of the order. For every $a \in Min(A)$ ($b \in Max(A)$) create a unique $p \in B$ with $p^\bullet = t(a)$ (${}^\bullet p = t(b)$).

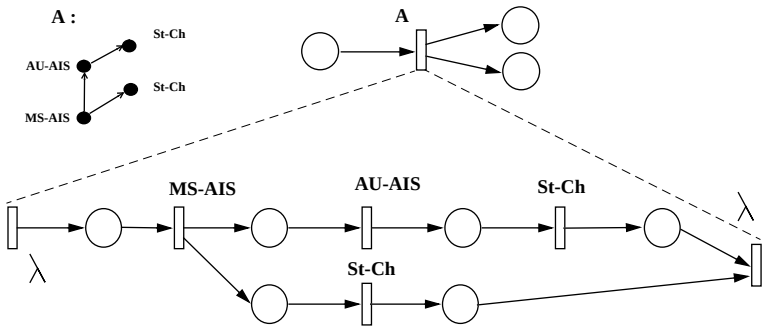

Figure 7: An alarm pattern and its corresponding labeled causal net

An alarm pattern is thus expanded onto a causal net, in which the expansion put for every $p \in P$, $|\ {}^\bullet p| = 1$ and $|\ p^\bullet| = 1$. The underlying partial order on transitions of such a causal net is exactly the partial order underlying the considered alarm pattern. This implies a bijection between alarms patterns and their corresponding labeled nets. To interface these expanded alarm patterns with places in the 1-safe nets, a λ-labeled transition is used as pre-set for the minimal (another one as post-set of the maximal) of the corresponding causal nets.

These λ-labeled transitions do not influence the semantics of the original labeled 1-safe net, because $\mathcal{S}_{oa}$ is defined according to partial orders on alarms.

To translate a Petri net of capacity one into a 1-safe net, simply add a complement to every place in the fault net. For a place $p \in P$ of the fault net, $\overline{p}$ is the complement of p if ${}^\bullet\overline{p} = p^\bullet$, $\overline{p}^\bullet = {}^\bullet p$ and $M_0(\overline{p}) = 1$ as initial marking. As Reisig marked out in [20], this is a convenient way to make the ability of a transition to fire dependent only on its pre-set. And the behavior of the considered net is left unchanged.

Figure 7 illustrates the expansion of an alarm pattern. On top of the figure an alarm pattern A is given, looking like one of the LOS fault net. A possible environment for a transition labeled A in a fault net is also proposed. The bottom of the figure illustrates the labeled net associated to this alarm pattern. In this example, complement of places are forgotten, to avoid clumsiness. It is clear that complements can be forgotten, because they do not influence either the behavior or the semantics of the fault net.

This leads to finite, 1-safe labeled system nets. Let $N = (P, T, F, M_0)$ such a net derived from a fault net, referenced as a 1-safe system net. This net is also T-restricted, i.e. $\forall t \in T,\ {}^\bullet t \neq \emptyset \neq t^\bullet$. To avoid problems due to labeling, places and transitions of the finite 1-safe net are numbered to distinguish any of them.

3.2.3 Family of Partial Orders on Alarms.

Partial order semantics of Petri nets is defined through the notion of *branching processes*. Branching processes are themselves based on a simple class of nets, *occurrence nets*.

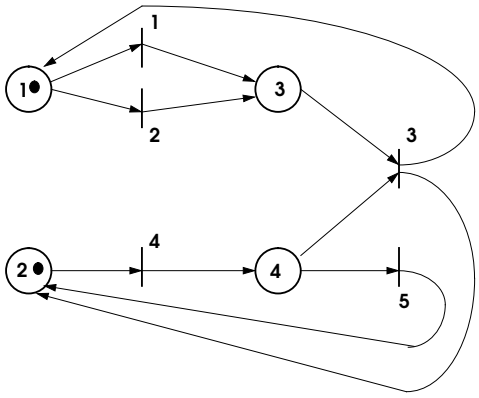

Figure 8: A 1-safe system net

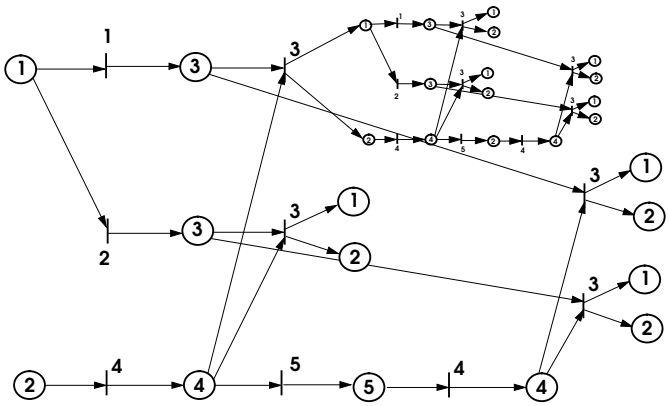

Figure 9: A branching process of the previous 1-safe system net

An occurrence net ON is a finitary acyclic net such that, for every $p \in P$, $| {}^\bullet p| \leq 1$ and $M_0 = Min(ON)$. Places are named conditions, and transitions are named events in occurrence nets.

A branching process of a net N is a pair (N', h) where N' is an occurrence net and h an homomorphism defined by the following mapping from N' to N, as in [6] for example.

Figure 8 gives an example of 1-safe system net. This example is chosen to illustrate unfolding, because it appears to highlight the behavior of Petri nets. Figure 9 contains a branching process of the previous 1-safe system net.

The *unfolding* of a Petri net is its unique maximal branching process [5].

A *prefix* (N', h') of (N, h) is defined as in [6].

Every branching process of a 1-safe system net can be seen as a prefix of its unfolding. Unfolding represents the infinite set of all these branching processes.

A *forwards conflict* appears when from the holding of $b \in B$ either $e_1 \in E$ or $e_2 \in E$, but not both, may occur (cf. Exclusiveness in Fig. 4). A *backwards conflict* appears when the holding of $b \in B$ may come from an occurrence of e_1 or e_2, but not both (cf. Or in

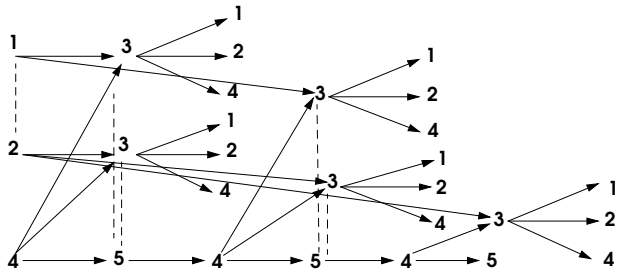

Figure 10: Event structure associated the previous branching process

Fig. 4). Conflicts are inherited by the dependency relation. This defines a symmetrical and irreflexive relation on E : $\#$, called the conflict relation, which satisfies : for every $e_1, e_2 \in E$, $e_1 \# e_2 \leq e_3$ implies $e_1 \# e_3$.

A *configuration* C of an occurrence net is a set of events satisfying: (i) $e \in C \Rightarrow \forall e' \leq e$: $e' \in C$ (C is closed for precedence), (ii) $\forall e, e' \in C, \neg(e \# e')$ (C does not contains conflict).

Now, given a 1-safe system net, are taken into account prefixes of unfolding whose set of events corresponds to a configuration.

As observations will be made of alarms and dependencies, only transitions have to be considered. This is why *event structures*, as defined in [22], are taken into account. Each occurrence net, and then branching process, defines an event structure [16].

The event structure associated to the unfolding of a 1-safe system net represents the set of event structures associated to the prefixes of its unfolding. It contains the infinite set of orders on events with conflicts.

The event structure associated to the branching process of Fig. 9 is provided by Fig. 10. The partial order relation is represented with black arrows, and direct forward conflicts are represented by vertical dashed lines.

The semantics of the fault net is the set of partial orders on alarms it described. Then, it is represented through the set of prefixes of the unfolding of the associated 1-safe system net.

$\mathcal{S}_{oa}$ *is the set of sub-orders underlying prefixes of this unfolding that correspond to configurations. It is contained in the event structure associated to the unfolding.*

Figure 11 illustrates a partial order on alarms, given by the fault net of Fig. 5. In this case, there is only a partial order on alarms corresponding to the fault net. The figure illustrates one of its sub-orders.

Remark: Orders on alarms described by a fault net are decomposable by series and parallel compositions on alarm patterns.

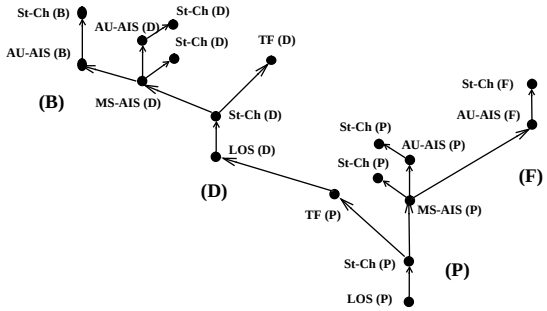

Figure 11: A partial order on alarms described by the previous fault net for LOS

4 Principle of Detection Algorithm

Typically, a manager receives lots of alarms, produced by elements of the data network, and has to correlate them to deduce the original fault(s). This section shows how this model is to be used to deal with this problem. It first details the hypothesis made about observation of alarms.

4.1 Alarm Observation

Each sensor of the management network observes alarms and some dependencies. Mechanisms like time-stamps provide a coding of dependencies that each sensor could observe between alarms ([8, 15]). Using this mechanism, it is easy to obtain, for each sensor, a partial order on alarms, with immediate predecessors known by the sensor for every alarm (causal observation [9]).

Figure 12 shows an example of observation provided by sensors for the case of loss of signal of STM-1 port P. Horizontal lines represents the time for each network element, for the sensors, and for the supervisor. The line of the supervisor illustrates the sequence of alarms provided to a manager without any information of concurrency between them. On the other hand, on the right hand of this figure, are the (Hasse) diagrams of observation provided by the sensors with time-stamps mechanism.

To avoid overloading the data network, no further intrusion in the traffic is allowed to observe dependencies. This would be too costly. For this reason, sensors do not exchange information about alarms. It implies that some dependencies between alarms could not be known: Inter-site dependencies are never observed by sensors. Each sensor only knows the intra-site dependencies it observes between alarms produced by elements of its site.

An observation provided to the operator is supposed to be in accordance with the effective order of production of alarms by the network elements, for each sensor.

Because of the non-observation of inter-site dependencies, alarms patterns can be simplified and decomposed according to sensors. For every alarm pattern, each sensor only keeps dependencies between alarms it observes. The transitive closure of the underlying partial

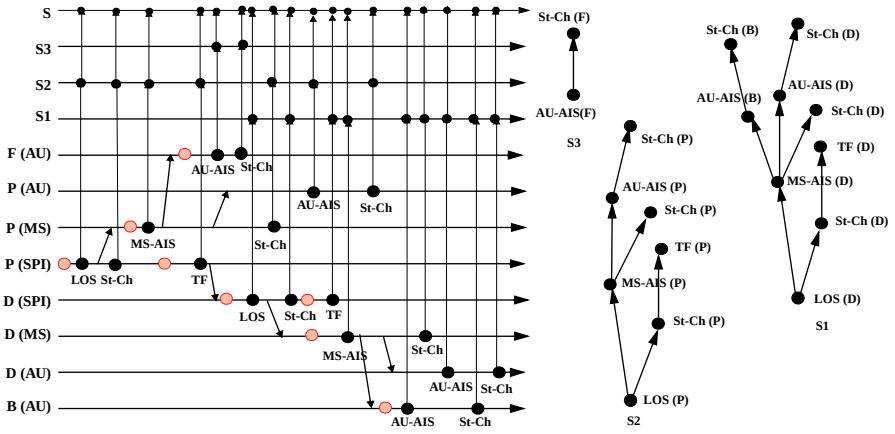

Figure 12: Ideal observation by sensors of alarms issued from a LOS

order of the considered alarm pattern is taken into account for this decomposition, to keep as much information as possible about dependencies from the original alarm pattern.

For a given alarm pattern A_i, the decomposition is denoted $\{A_i s\}$ where s enumerates sensors concerned with alarms of A_i. For two alarms x, y of A_i:

1. $x \leq y$ (there is an expected dependency between x and y);
2. $s(x) = s(y)$ (they are observed by the same sensor);
   (a) if $x \prec y$ in A_i, then $x \prec y$ in $A_i s$;
   (b) if there is at least an alarm z in A_i, with $s(z) \neq s(x)$, such that $x \leq z \leq y$ in A_i but no other alarm a, with $s(a) = s(x)$, such that $x \leq a \leq z \leq y$ nor $x \leq z \leq a \leq y$ then $x \prec y$ in $A_i s$.

Only alarm patterns are decomposed. The whole structure of the net is to be known by each sensor.

4.2 Incremental Construction of Unfoldings

The question of detection is : given partial orders on alarms provided by sensors, which histories of faults did happened ? i.e., which are the partial orders of $\mathcal{S}_{oa}$ that explain the partial orders given by sensors ?

There are in general several solutions, because of non-observed dependencies, because different alarm patterns can have similar alarms, and also because alarms can be lost.

In [17], the unfolding is shown to be equivalent to the union of the processes of a 1-safe net, with conflicts between these processes. Solutions, i.e. elements of $\mathcal{S}_{oa}$, are thus obtained through construction of processes. Furthermore, in case of 1-safe system nets, there is a one-to-one correspondence between processes and orders on events. These good properties of 1-safe system nets are useful to build diagnosis.

A causal net is a finite Petri net such that, for every $p \in P$, $|\ {}^\bullet p| \leq 1$ and $|\ p^\bullet| \leq 1$.

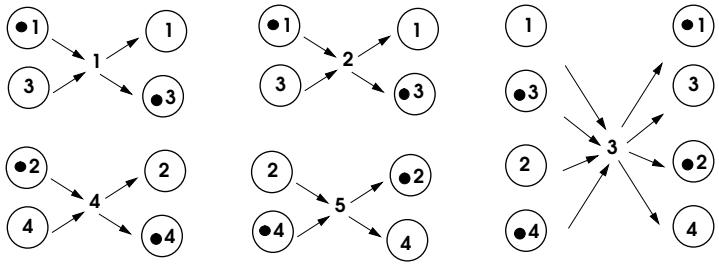

Figure 13: Tiles used for construction of processes

The construction of processes is considered like a puzzle, with tiles extracted from the net. To avoid clumsiness, we choose to use capacity one nets firing rules, instead of having double number of places. So, each tile simulates the firing of a transition, according to capacity one nets firing rules.

A *tile* is centered on a transition $t \in T$ of the Petri net. It has a set of pre-conditions corresponding to required value of tokens to enable the fire of t; and a set of post-conditions, corresponding to value of tokens given by the firing of t. Tiles are directly extracted from the structure of the given system net. Figure 13 illustrates these tiles used for the construction of processes for system net of Fig. 8.

Remark: In this net, all places already have a complement.

4.3 Principles of Processes Construction

The considered way of construction of processes consists in the following. One enumerates all possibilities for tiles to be put at a given step of the construction. Each possibility, after having put the tiles, leads to a new unfolding.

To put a tile, one just has to superimpose its pre-conditions to existing post-conditions in the already done unfolding. A tile can be put only if the unfolding preserves the observed dependencies.

Detection Algorithm The recognition algorithm is like a construction of processes associated to the net derived from the given fault net, but constrained by the observed alarms and dependencies. It was first inspired from [2], and next from [6].

Initialize a process with initially marked places of the net
For each observed event e
  For each tile T_e centered on a transition labeled e
    For each current process S
      enumerate(T_e,S) /*set of possible places σ in S */
    end /*S*/
  end /*T_e*/
  For each current process S

```
        if there is one σ for in S
          S := put(σ,S)
        else for each possible σ
          S' = S /*duplicate S */
          S'' := put(σ,S')
          S'' is inserted in the set of processes
        end /*if*/
    end /*S*/
end /*e*/
```

enumerate(P_e,S):

This enumeration is done according to the capacity one net firing rules. This procedure takes into account λ-labeled transitions whose pre-conditions are in S (as immediately recognized events).

put(P_e,σ,S):

This procedure takes into account immediate predecessors of e in the given observation, and ensure that they correspond to predecessors (not λ-labeled) in S.

The more subtle part of this algorithm is to deal with λ-labeled transitions.

Figure 14 give a possible labeling of transitions of the net of Fig. 8. Alarms labeling transitions are in parenthesis for labeled transition (1,2,5). On top of Fig. 15 is an example of observed alarms, with their dependencies. Below is the process constructed to recognized this order with the labeled net. The process is represented with an horizontal line for tokens of each place. Tiles are put with transitions on bottom of the figure. Labels of transitions are given when existing, otherwise the name of the transition is reported. Transitions are linked to their pre- and post-conditions by oblique arrows.

This example is chosen to show the influence of both the constraint on predecessors and λ-labeled transitions. Suppose that $a1$ is recognized, i.e. tile centered on $a1$ is already put. Now, suppose b is to be recognized.

It seems to be a solution to build a new process by putting the tile centered on transition 4 and then the tile centered on transition 5, that is labeled b. Unfortunately, this process does not preserve the observed dependency between $a1$ and b. It would be a solution if $a1$ and b were independent.

Here, the λ-labeled transition 3 *must* be fired to ensure that $a1$ precedes b in the constructed process. And for tile centered on b to be put, place 4 must be marked, and place 2 unmarked. But this example exhibits that there are no hypothesis about marking of other places, 1 and 3. So, it is necessary to enumerate all markings of the net that contain require conditions to fire transition 5, they are named here final markings.

This implies then a backward searching on possible λ-labeled transitions to be fired. This backward searching begin with final markings, and goes back to reach the current max of already constructed process(es). It stops when no λ-labeled transition allows to go backward any more or when a cycle appear. While going backwards through λ-labeled transitions,

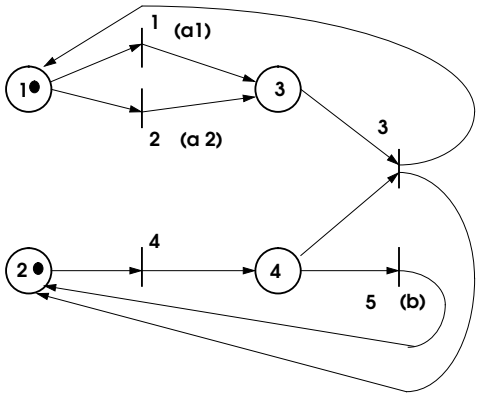

Figure 14: Labeled 1-safe vs. capacity-one system net

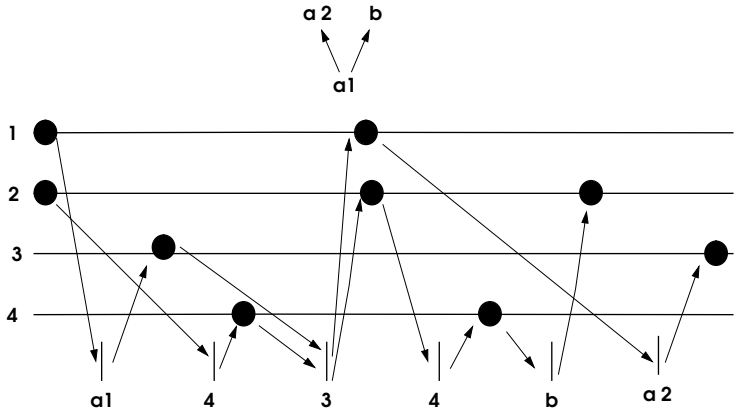

Figure 15: Observed alarms and their dependencies, and process constructed to recognize this order

direct backwards conflicts can not be resolved. Those conflicts generate as many solutions as the number of choices they imply.

And, in final, this has to be done in such a way that predecessors agrees to put the tile. So, only processes whose order on labeled transitions agree with the observed dependencies are kept, and others are discarded.

4.4 Implementation

This algorithm is implemented in Eiffel language, which is object oriented. It takes about three to four months to develop this algorithm. The commented code takes approximatively 10 000 lines for 22 000 words.

In this algorithm, there are two types of non-determinism. Alarms could be ambiguous. For example, suppose that transitions 1 and 2 of the previous labeled net where both labeled with a, instead of having different labels $a1$ and $a2$. And now, suppose we want to recognized the observed order with two a, instead of observed $a1$ and $a2$. There would be four solutions instead of only one, because, for every observed a, tiles centered on transitions 1 and 2 are

possible.

And λ-labeled transitions introduce non-determinism that can not be reduced only by the given observation, as explained above.

However, in practice, these non-determinism should be limited. The case of ambiguous alarms should not be too frequent, even if we did not want to impose an hypothesis such that a unique naming of alarms. Next, there must not be too much fault states chosen without any alarm manifestation, even if this allows more acuteness in fault states dependencies. λ-labeled transitions may appear in major party because of alarms patterns expansion. This could introduce backwards direct conflicts between λ-labeled transitions, but these conflicts are quickly resolved with alarms of patterns.

The algorithm is to be distributed and sensors, and parallelized in supervisor.

Every sensor, as designed in Fig. 1, should know the entire structure of the fault net, and the part of alarm patterns it is concerned with. Thus, each sensor could compute separately the enumeration of possible tiles for an alarm it observes, in every current process.

In the scheme of the algorithm, it is easy to see that the step " **For each** current process " could be parallelized inside the supervisor. Thus a supervisor can build in parallel different solutions, by putting tiles. Then, the supervisor gives to the sensors the new built processes.

5 Related Works

As far as we know, some other works have common aspects with the work presented in this article.

Petri nets has already been used for diagnostic, see e.g. [7] in the field of automation. They establish a linear logic associated to Petri net. The linear negation is used to diagnose faults. Petri nets are also used for automatic diagnosis in the field of artificial intelligence. As mentioned in [19], we use Petri nets to exploit the parallelism of concurrent evolutions. But we do not represent the whole behavior of the system to be diagnosed. And we take advantages of the partial order semantics of 1-safe Petri nets, rather than using reachability or T-invariant analysis.

In [21] alarm correlation is considered by the way of database operations. Events are represented by data-patterns, and data patterns are continuously retrieved from the network database. Events are correlated through combination of events. By correlation rules: disjunctions, conjunctions; by temporal relationships: temporal order, by the way of an acyclic directed graph, and temporal constraint, using occurrences dates. In our approach, we do not want to use real time, in order to avoid problems due to clock synchronization. We think that most of aspects dealing with real time mask causal dependencies and concurrency, which we explicit, both at the model level (fault net) and at the observation level (time-stamps).

In [11], alarm correlation is a conceptual interpretation of multiple alarms, such that a new meaning is assigned to these alarms. Their approach is based on the principles of model-based reasoning. Dependencies between faults are represented by logical fault propagation rules: causal implication, and, or. In [12] they concentrate on time-dependent events management.

In our model, a great variety of dependencies between faults and alarms can be expressed. In particular, because they are managed in a uniform manner, we take into account the problem of fault inter-dependencies due to alarm propagation.

In [4], two finite states machines are used. One models the whole behavior of the considered system, that is partially observed by the second. Faults are considered as changes and additions of arcs in the finite state machine of the system. Therefore, the detection problem is expressed in terms of sequences generated (or not) by the system. The focus of their work is in the design of the simplest observer for a given finite state machine. Our propagation model does not describe the whole behavior of the system, but rather its manifestations in case of malfunctions.

In [14], problems are viewed as messages encoded in the set of alarms they cause. By pruning, a bipartite correlation graph is obtained. It provides for each problem a code by the way of a vector of alarm. Then the alarm correlation problem is that of finding problems whose code optimally match an observed alarm vector. The way to perform matching is left open. For a more general class of models, this is what our algorithm does.

In [13] probabilistic methods are also used, but without model of concurrency between events.

6 Concluding Remarks and Future Works

This paper takes advantages of results in the theory of Petri nets to deal with fault management in telecommunication network. It takes explicitly into account multiplicity and different causal dependencies between faults. And it allows to express dependencies and concurrencies between alarms.

As far as we know, this model is a new approach to fault detection through alarm correlation. The choice of partial order semantics of Petri nets prevents from the combinatorial explosion of interleaving when dealing with concurrency. The use of time-stamp mechanism prevents from noise due to physical clock distortion of observation.

This model is illustrated with an example of SDH network. The detection algorithm is programmed with the Eiffel language, and we plan real experiments.

A joined work with the use of probabilities [1] completes this approach, in order to propose a new complete way to diagnose faults in telecommunication network.

A short term research is to define precisely the way to provide diagnosis (i.e. explanation) and to be able to distribute the supervising activity on sensors:

1. A diagnostic could be provided to the operator like a backward simulation of the fault net according to the results given by the recognition. This allows the operator to go back to the original fault through the fault net. The idea of prefix used to define $\mathcal{S}_{oa}$ is linked to the notion of correlation window. Its size has to be determined by the network operator, according to a number of alarms, or to a number of fault states, or maybe to a delay.

2. All this work is done keeping in mind the aim of distributed detection. This detection algorithm can be distributed over sensors. Those sensors are able to process part of the

recognition, with the set of alarms and their dependencies they observe respectively. The decomposition of alarm patterns on sensors is then very useful. Then, the task of the supervisor is to synchronize results from sensors to compute a diagnosis. The supervisor could also give global trends indications to sensors, and compute the different solutions in parallel.

References

[1] A. Aghasaryan, E. Fabre, A. Benveniste, R. Boubour, and C. Jard. A Petri Net Approach to Fault Detection and Diagnosis in Distributed Systems. part II: Extending Viterbi Algorithm and Hmm Techniques to Petri Nets. To appear in IEEE Conference on Decision and Control, december 1997.

[2] E. Best and R. Devillers. Sequential and Concurrent Behaviour in Petri Nets Theory. *Theoretical Computer Science*, (55):87–136, 1987.

[3] R. Boubour and C. Jard. Une approche pour des capteurs d'alarmes intelligents dans les réseaux. In A. Bennani, R. Dssouli, A. Benkiran, and O. Rafiq, editors, *CFIP'96 : Ingénierie des Protocoles*, octobre 1996.

[4] A. Bouloutas, G. Hart, and M. Schwartz. *On the Design of Observers for Fault Detection in Communication Networks*, chapter 5. New-York : Plenum, 1990.

[5] J. Engelfriet. Branching Processes of Petri Nets. *Acta Informatica*, 28(6), 1991.

[6] J. Esparza, S. Römer, and W. Volger. An Improvement of McMillan Unfolding Algorithm. In *2nd Int. Workshop TACAS*, number 1055 in Lecture Notes in Computer Science. Springer-Verlag, march 1996.

[7] R. Valette et L.A. Künzle. Réseaux de Petri pour la détection et le diagnostic. In *Journées Sûreté, surveillance, supervision : détection et localisation des défaillances*, number LAAS-R-94463, novembre 1994.

[8] J. Fidge. Timestamps in Message Passing Systems that Preserve the Partial Ordering. In *Proc.* 11^{th} *Australian Computer Science Conference*, pages 55–66, February 1988.

[9] E. Fromentin, C. Jard, G.-V. Jourdan, and M. Raynal. On-the-fly Analysis of Distributed Computations. *Information Processing Letters*, (54):267–274, 1995.

[10] John M. Griffiths, editor. *ISDN Explained : Worldwide Network and Applications Technology*. John Wiley and Sons, 1990.

[11] G. Jakobson and M. D. Weissman. Alarm Correlation. *IEEE Network*, 7(6), November 1993.

[12] G. Jakobson and M. D. Weissman. Real-time Telecommunication Network Management : Extending Event Correlation with Temporal Constraints. In Sethi, Raynaud, and Faure-Vincent, editors, *Integrated Network Management*, number 4, pages 290–301. IFIP, Chapman and Hall, may 1995.

[13] I. Katzela, A.T. Bouloutas, and S.B. Calo. Centralized vs Distributed Fault Localisation. In Sethi, Raynaud, and Faure-Vincent, editors, *Integrated Network Management*, number 4, pages 250–261. IFIP, Chapman and Hall, may 1995.

[14] S. Kliger, S. Yemini, Y. Yemini, D. Ohsie, and S. Stolfo. A Coding Approach to Event Correlation. In Sethi, Raynaud, and Faure-Vincent, editors, *Integrated Network Management*, number 4, pages 266–277. IFIP, Chapman and Hall, may 1995.

[15] F. Mattern. Virtual Time and Global States of Distributed Systems. In Cosnard, Quinton, Raynal, and Robert, editors, *Proc. Int. Workshop on Parallel and Distributed Algorithms Bonas, France, Oct. 1988*. North Holland, 1988.

[16] M. Nielsen, G. Plotkin, and G. Winskel. Petri Nets, Event Structures and Domains, part 1. *Theoretical Computer Science*, (13), 1981.

[17] M. Nielsen, G. Rozenberg, and P.S. Thiagarajan. Behavioral Notions for Elementary Nets Systems. *Distributed Computing*, 4, 1990.

[18] Y.A. Nygate. Event Correlation using Rule and Object Based Techniques. In Sethi, Raynaud, and Faure-Vincent, editors, *Integrated Network Management*, number 4, pages 278–289. IFIP, Chapman and Hall, may 1995.

[19] L. Portinale. Exploiting T-invariant Analysis in Diagnostic Reasoning on a Petri Net Model. In *Procedings of Application and Theory of Petri Nets 1993*, number 691 in Lecture Notes in Computer Science. Springer-Verlag, 1993.

[20] W. Reisig. *A Primer in Petri Nets Design*. Springer-Verlag, 1992.

[21] O. Wolfson S. Sengupta and Y. Yemini. Managing Communication Network by Monitoring Databases. *IEEE Transactions on Software Engineering*, 17(9), Septembre 1991.

[22] W. Vogler. *Modular Construction and Partial Order Semantics of Petri Nets*. Number 625 in Lecture Notes of Computer Science. Springer-Verlag, 1992.