Extending Decentralized Discrete-Event Modelling to Diagnose Reconfigurable Systems

Alban Grastien[1] and **Marie-Odile Cordier**[1] and **Christine Largouët**[2]

Abstract.
On-line reconfiguration is the ability to rearrange dynamically the elements of a system to accommodate failure events or new requirements. Due to the modular representation, decentralized discrete-event approach, recently proposed for the diagnosis of systems, is particularly well suited to the diagnosis of reconfigurable systems. The contribution of this article is to extend our decentralized approach to reconfigurable discrete-event systems. A first step in this direction is to extend the way a decentralized system is modelled. The idea consists in modelling separately the behavior of the components and the system topology. A second step is to formally define what is a reconfiguration. A property of reconfiguration, that we call *safety*, is identified to be important. When satisfied, we show that our decentralized diagnosis approach can easily be extended to reconfigurable systems.

1 INTRODUCTION

Real-world decentralized systems are designed to enable reconfiguration, *i.e.*, the modification of the connections between the components and/or the addition or removal of components. This is particularly the case when those systems are networks of components such as telecommunication or power transportation networks. The reason of a reconfiguration can be the update of the system (substitution/addition of components) or an emergency procedure to protect the system from a failure in a subsystem (removal of connections). Another benefit of on-line reconfiguration arises from its possible integration with diagnosis [5]. Thus a relevant reconfiguration can be chosen to refine the discrimination between diagnosis and then to gain amount of time and efficiency in finding the right fault.

In those examples, it is clear that reconfiguring a system should not stop the diagnosis task, even if it is done on-line. However, most of the diagnosis approaches are topology-dependant, as for example expert systems or chronicle-based systems [3]. Model-based diagnosers [11, 10] are also generally unable to deal with this task since they rely on a global system model which either is too large if it accounts for all possible topologies, or too costly to compute on-line if the model has to be changed during the diagnosis task. Due to the great number of topologies of highly reconfigurable systems, it is thus not reasonable to rely on an explicit global model especially when the model of future components is not necessarily known at the time of its construction.

Decentralized approaches, as presented in [7, 9], are interesting since they do not require to compute an explicit global model and consider a system as a set of connected components. Successfully used for diagnosis, they appear thus well suited for on-line reconfiguration because of their modular architecture: on the fly computation of local diagnoses is flexible enough to add or remove components in the system. However, until now it is generally assumed that the topology of the system does not change on-line.

In this paper, we extend the decentralized approach in [9] to reconfigurable discrete-event systems. The reconfiguration actions are decided by an operator that informs the diagnosing system which therefore knows exactly the topology of the supervized system. A first step in this direction is to extend the way a decentralized system is modelled. The idea is to model separately the behavior of the components and the system topology. The notion of topology is formally introduced and defines the connections between the components. It becomes possible to change it in a modular way. Even if not explicited, the system model is then well-defined as the synchronization of models of connected components. A second step is to formally define a reconfiguration[3]. A property of reconfiguration, that we call *safety*, is identified to be important. When satisfied, we show that the decentralized diagnosis approach proposed in [9] can easily be extended to reconfigurable systems.

This paper is organized as follows. In Section 2, we present an illustrative example that we use throughout the whole paper. Section 3 introduces the modelling of reconfigurable systems by stating some simplifying hypotheses. Section 4 defines reconfiguration and presents the safety property. Section 5 illustrates our contribution by examining successive reconfigurations on a running example. Finally, the method used for taking into account reconfigurable systems by the decentralized diagnosis approach is sketched in Section 6.

2 AN ILLUSTRATIVE EXAMPLE

In this section we present an example of a system that support different configurations. The device is composed of a pump P and two pipes PI1 and PI2. The pump delivers the water while the pipes carry it to other components (out of the studied system).

The pump has three modes of behavior: the *OK* behavior and two faulty behaviors (*leaking* and *blocked*). Firstly the pump can leak (fault $f1$) and then the output flow becomes low. Secondly, the pump can block ($f2$) and the output flow is null. In each mode, the pump can be stopped (action *off*) and started again (action *on)* by an operator. During an action *on* or *off*, the mode of the pump is not changed.

The pipes exhibit two modes of behavior, the *OK* behavior (all the received water is delivered to the output) and a faulty behavior (fault

[1] Irisa, Université de Rennes 1, Campus de Beaulieu, 35042 Rennes Cedex, France, {agrastie,cordier}@irisa.fr
[2] University of New Caledonia, BP. 4477, 98847 Nouméa Cedex, New Caledonia, largouet@univ-nc.nc

[3] The way this reconfiguration is chosen is out of the scope. Such design problems are discussed in [12] for example.

F) when the pipe is *leaking*: in this case, all or part of the input flow is lost.

The output flow of each component can be high (denoted by h), low (denoted by l) or zero (denoted by z).

As will be seen later in the models (subsection 3.1), two non-deterministic cases are considered for a pipe when it is leaking and when it receives a low flow (from the pump or from the other pipe). Its output flow is lower than its input flow. We consider that it can be measured either as a low or zero flow, according to the importance of the leaking.

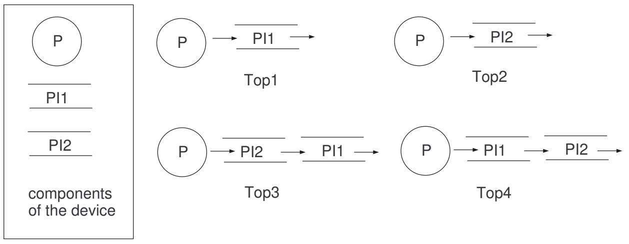

Figure 1. Possible topologies of the system

Our system delivers water to an external component. The modularity of the system enables us to use it for different tasks. For example, the pump can be used with a unique pipe (PI1) as depicted in the topology $Top1$ of Figure 1. If a leak occurs on PI1, PI1 can be replaced by the pipe PI2 as depicted in the topology $Top2$. If a new functionnality is required, the second pipe is connected, as depicted by the topologies $Top3$ or $Top4$, at the end of the first one. Those evolutions of topology are called reconfigurations.

Some reconfiguration actions are not allowed in some states of the system. For example, the action of connecting a pipe to the pump cannot be realized while it is delivering water. It is first necessary to stop the pump.

Considering an on-line diagnosis task as monitoring the flow out of a pipe, the on-line reconfiguration should not stop the diagnosis task and it should take into account the evolution of the topology in the model.

3 MODELLING RECONFIGURABLE SYSTEMS

This section concerns the modelling of reconfigurable systems. It must first be noted that, since physical components (or connections between them) can be modified on-line, the decentralized way of modelling a system as presented in [7, 9] is adequate for reconfigurable systems. The decentralized model proposed in [9] is consequently extended in order to allow a more precise description of the way components are physically connected by connection points (or ports). In the following, we successively examine the component model, the topology model and the system model. To simplify the presentation, we make some hypotheses which are given along the text.

3.1 Component model

A *component* is a (physical or abstract) element. Each component may have *connection points* (sometimes called *points*). Each point may be linked by a *connection* to a point of another component (the former point is called internal point) or to the system environment (external point).

Communications (flow, messages, *etc.*) between components are made through connections. By abuse of language the content of a communication is called a *message*. A message is said to be *internal* when it is sent by another component. It is said to be *exogeneous* when it is sent by the system environment. It can be supposed, without loss of generality, that no more than one exogeneous message can be received by a component at the same time. Consequently, we have the following hypothesis:

Hypothesis 1 *Each component has a unique connection point with the environment. This only external point is noted p^{ext}.*

A component is modelled as a discrete-event system, which means that its state is only changed on the reception of a message. As usual, we make the following assumption:

Hypothesis 2 *A component can receive only one (internal or exogeneous) message at the same time.*

The component behavior is described by a finite state machine Σ.

Definition 1 (Component Model) *The model of the component is described by the finite state machine $\Sigma = \langle Q, E, P, T, Q^o\rangle$ where:*

- *Q is the set of component states,*
- *E is the set of messages,*
- *P is the set of (connection) points, $p^{ext} \in P$ is the external point,*
- *$T \subseteq (Q \times (P \times E) \times (P \times E)^\star \times Q)$ is the set of transitions,*
- *Q^o is the set of initial states.*

A transition $t = (q,\ (p,e),\ \{(p_1,e_1),\dots,(p_k,e_k)\},\ q')$ can be read as follows: in the state q, the component receives the message e on the point p. Then, it goes in the state q' and emits the messages e_i on the points p_i ($i \in \{1,\dots,k\}$).

In order to allow reusability, two or more components may have the same component model. A component c_k is associated to its model by the function Mod, where $Mod(c_k) = \Sigma_k$, an instance of a component model Σ. The difference between two different instances of Σ, Σ_i and Σ_j, is that labels are distinct and indiced by respectively i and j.

To illustrate the definition of component model, the models of a pump and of a pipe are given in Figure 2 and Figure 3. The transition label $p\!:\!e|\{(p_i\!:\!e_i)\}$ means that the transition is triggered by the reception of a message e at the connection point p and that each message e_i is sent to the connection point p_i.

3.2 Topology model

As seen before, a component is modelled as a discrete-event system, which means that its state changes on the reception of a message and that the component reacts by sending messages. It is then clear that the behavior of connected components is constrained by the way they communicate. This constraint is called *synchronization* and is described by a *synchronization set*.

Definition 2 (Synchronization set) *A* synchronization set*, denoted $|$, between two models of components $\Sigma_1 = \langle Q_1, E_1, P_1, T_1, Q_1^o\rangle$ and $\Sigma_2 = \langle Q_2, E_2, P_2, T_2, Q_2^o\rangle$ is a subset of the pairs of messages of the two finite state machines: $| \subseteq E_1 \times E_2$.*

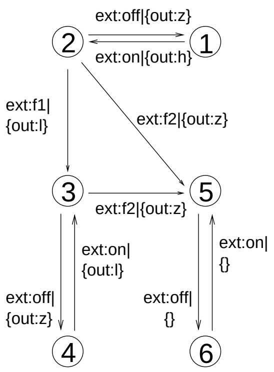

Figure 2. Model of a pump

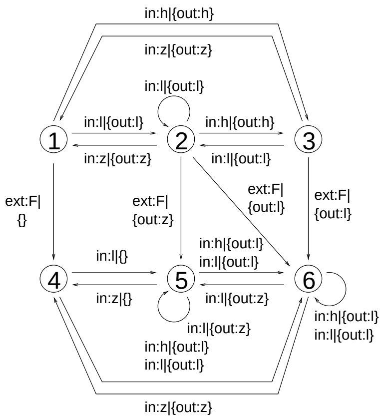

Figure 3. Model of a pipe

In our example, the synchronization set indicates that the flow emitted by the pump is connected to the input flow of the pipe (for example, a low flow in output of the pump corresponds to a low flow in input of the pipe). The synchronization set is then the identity function, *i.e.*, $\{(h, h), (l, l), (z, z)\}$.

A connection is described by the connected points and the synchronization set which has to be satisfied.

Definition 3 (Connection) *A connection co is defined as a n-uplet* $((c_1, p_1), (c_2, p_2), |)$ *such that:*

- $c_1 \neq c_2$,
- $\forall k \in \{1, 2\}, \Sigma_k = \langle Q_k, E_k, P_k, T_k, Q_k^o \rangle = Mod(c_k), p_k \in P_k$,
- $|$ *is a synchronization set between* $Mod(c_1)$ *and* $Mod(c_2)$.

By definition, a component cannot be connected to another component by its external point: $\forall k \in \{1, 2\}, p_k \neq p_k^{ext}$.

The set of connections in the system is called the *topology* (or the *configuration*).

Definition 4 (Topology model) *The topology model of a system* Top *is a set of connections between the components of the system such that no point is connected more than once:* $\forall co, co' \in Top$, $co = ((c_1, p_1), (c_2, p_2), |)$, $co' = ((c'_1, p'_1), (c'_2, p'_2), |')$, $co \neq co'$, $\forall i \in \{1, 2\}, j \in \{1, 2\}$, $c_i = c'_j \Rightarrow p_i \neq p'_j$.

3.3 System model

The model of the system depends on the model of each of its components and of the topology model. We first present simplifying hypotheses. Then, the decentralized model of the system and the explicit behavior it describes are defined. It can be noticed that these definitions are direct extensions of the definitions which are given in [9]. The main difference is the addition of the explicit topology of the system which was only implicit in [9]. The explicit description of the topology is necessary when dealing with reconfigurable systems. It can also be noted that this topology model is a simplified form of the link models proposed in [7].

3.3.1 Simplifying hypotheses

The system is modelled as a discrete-event system which means that it evolves on the occurence of exogeneous messages. We make the following hypothesis:

Hypothesis 3 *The system can receive only one exogeneous message at the same time.*

The next hypothesis states that we focus on synchronous systems. To consider communications with delays and/or losses during the transmission of messages in our system, the communication channel should be modelled as a component connected to the components it connects.

Hypothesis 4 *Communications between components are instantaneous.*

As said previously, components are modelled as discrete-event systems, which means that, when a component receives an exogeneous message, it may react by sending messages to other components, which may themselves react. This propagation of messages has to satisfy some finiteness properties, which explains the following hypothesis:

Hypothesis 5 *The propagation of a message through the components can be described by a tree of visited components. In this tree, a component can only be visited once.*

3.3.2 Decentralized model of the system

Definition 5 (Decentralized Model of the System) *A* decentralized model of the system *is a n-uplet* $(Comp, Mod, Top)$ *where:*

- $Comp$ *is the set of components of the system,*
- Mod *maps each component to its model,*
- Top *is the topology of the system.*

3.3.3 Explicit behavior of the system

The decentralized model of the system completely defines the behavior of the system. This behavior is implicit, but can be explicitly computed as follows.

We introduce the notion of ε-transition, which corresponds to the fact that no message is received by a component. The state of the component is not modified by an ε-transition.

Definition 6 (Free product) *Let* $\Sigma_i = \langle Q_i, E_i, P_i, T_i, Q_i^o \rangle$ *be the models of the* n *components* c_i*, the* free product *of* n *finite state machines* Σ_i *is a finite state machine* $\Sigma = \langle Q, E, P, T, Q^o \rangle$ *where:*

- $Q = Q_1 \times \cdots \times Q_n$,
- $E = E_1 \cup \cdots \cup E_n$,
- $P = P_1 \cup \cdots \cup P_n$,
- $T = (T_n \cup \{\varepsilon\}) \times \ldots \times (T_n \cup \{\varepsilon\})$ *is the set of transitions* $(q_1, \ldots, q_n) \xrightarrow{(m_1,\ldots,m_n)} (q'_1, \ldots, q'_n) = (q_1 \xrightarrow{m_1} q'_1, \ldots q_n \xrightarrow{m_n} q'_n)$*, where* $q_i \xrightarrow{m_i} q'_i$ *is a transition of* T_i *or an* ε*-transition,*
- $Q^o = Q_1^o \times \ldots \times Q_n^o$.

The free product computes the behavior of the system without any constraint on the connections.

Let t be a transition $(t_1, \ldots, t_n)$, with $t_i = (q_i, (p_i, e_i), \{((p_{i,1}, e_{i,1}), \ldots, (p_{i,k}, e_{i,k}))\}, q'_i)$ or $t_i = \varepsilon$. We note $t = (q, eed, eeg, q')$, where $q = (q_1, \ldots, q_n)$, $q' = (q'_i, \ldots q'_n)$, eed is the set of the messages received by the components and eeg the set of the messages sent by the components. They are computed by these formulæ: $eed = \bigcup_i \{(p_i, e_i)\}$ and $eed = \bigcup_{i,j} \{(p_{i,j}, e_{i,j})\}$ ($\forall i$ such that t_i is not an ε-transition).

Definition 7 (Synchronization on a connection) *A transition $t = (q, eed, eeg, q')$ is synchronized on a connection $((c_j, p_j), (c_k, p_k), |)$ if:*

- $(\exists e_j, (p_j, e_j) \in eeg) \Rightarrow (\exists e_k, (p_k, e_k) \in eed \wedge (e_j, e_k) \in |)$,
- $(\exists e_k, (p_k, e_k) \in eed) \Rightarrow (\exists e_j, (p_j, e_j) \in eeg \wedge (e_j, e_k) \in |)$.

The synchronization on a connection checks that any emitted message is received and conversely.

Definition 8 (Synchronization on a topology) *A transition $t = (q, eed, eeg, q')$ is synchronized on the topology Top of the system if:*

- *it is synchronized on each connection of the topology,*
- $\forall (p, e),\ (p, e) \in eed \Rightarrow (\exists c_1, p_1, c_2, |,\ ((c_1, p_1), (c_2, p), |) \in Top) \vee (\exists i, p = p_i^{ext})$,
- $\exists! p, (\exists i, p_i^{ext} = p) \wedge (\exists e, (p, e) \in eed)$

The first proposition of Definition 8 ensures that the messages are synchronized on the connections. The second proposition ensures that every received message belongs to a connection or is received on an external point. Note that a message can be sent even if no component received it when the connection point is deconnected. The third proposition ensures that a transition contains exactly one exogeneous message.

Definition 9 *The explicit behavior of the system $(Comp, Mod, Top)$ is the finite state machine $\Sigma' = \langle Q', E, P, T', Q^o \rangle$ from the free product $\Sigma = \langle Q, E, P, T, Q^o \rangle$ such that $Q' \subseteq Q$ is the set of states and $T' \subseteq T$ is the set of synchronized transitions of E.*

4 RECONFIGURING DISCRETE EVENT REACTIVE SYSTEMS

A *system* is a network of connected components and the set of current connections is called the *topology* (or the *configuration*). The modification of a topology is called a *reconfiguration*. The addition of a new connection is called a *connection action* while the removal of a connection is called a *deconnection action*. A system is said to be *reconfigurable* when its topology may be reconfigured.

Definition 10 (Deconnection action) *A* deconnection action a_d *removes a connection from the system. The deconnection action is denoted by the connection that is removed: $a_d = ((c_1, p_1), (c_2, p_2), |)$.*

Definition 11 (Connection action) *A* connection action a_c *adds a connection to the system. The connection action is denoted by the connection that is added: $a_c = ((c_1, p_1), (c_2, p_2), |)$.*

A reconfiguration is a set of connection and deconnection actions. We consider that these actions are instantaneous. It means that no event can occur between two actions.

Definition 12 (Reconfiguration) *A* reconfiguration $\mathcal{R}$ *in a topology Top is a pair $(DA_{\mathcal{R}}, CA_{\mathcal{R}})$ where $DA_{\mathcal{R}}$ is a deconnection action set ($DA_{\mathcal{R}} \subseteq Top$) and $CA_{\mathcal{R}}$ is a connection action set, such that: $\mathcal{R}(Top) = (Top \setminus DA_{\mathcal{R}}) \cup CA_{\mathcal{R}}$ is a topology.*

$\mathcal{R}(Top)$ is the topology of the system after the reconfiguration.

We consider that a reconfiguration cannot be executed in any state of the system components. For example, it is clearly not safe to disconnect the output of a pump while it is delivering water. More precisely, the safety of a reconfiguration depends on the connection points concerned by the reconfiguration. For instance, we can imagine a pump with two connection points, one connected with a pipe and the other with a container; the pump has to be stopped when connected to a new pipe but has not to be stopped when connected to a new container.

It is why, before formally defining the reconfiguration safety, we extend the component model and associate to each connection point the set of states in which a reconfiguration is allowed.

The component model is extended by the addition of G as follows:

Definition 13 (Extended model of a component) *The model of the component c is described by the finite state machine $\Sigma = \langle Q, E, P, T, G, Q^o \rangle$ where:*

- *Q is the set of component states,*
- *E is the set of messages,*
- *P is the set of (connection) points, $p^{ext} \in P$ is the external point,*
- *$T \subseteq (Q \times (P \times E) \times (P \times E)^\star \times Q)$ is the set of transitions,*
- *$G \in (P - \{p^{ext}\} \rightarrow 2^Q)$ is a function that associates to each internal connection point the set of states that support a reconfiguration,*
- *Q^o is the set of initial states.*

In our example, the set of states in which the pump (see Figure 2) may be reconfigured, is given by $G_P(out) = \{1, 4, 5, 6\}$ (out is the only internal connection point). These states are those where there is no outflow. For the pipe (see Figure 3), which has two internal connection points, we have $G_{PI}(in) = \{1, 4\}$, and $G_{PI}(out) = \{1, 4, 5\}$, 1 and 4 being states where there is no inflow, and 1, 4 and 5 being states where there is no outflow.

A *safe* reconfiguration is defined as follows:

Definition 14 (Safe reconfiguration) *A reconfiguration $\mathcal{R}$ is* safe *if $\forall ((c_1, p_1), (c_2, p_2), |) \in DA_{\mathcal{R}} \cup CA_{\mathcal{R}}$, the current state of the component c_i is in $G_i(p_i)$, $\forall i \in \{1, 2\}$.*

The safety property of a reconfiguration ensures that the state of any component is not changed by the reconfiguration. For instance, it seems normal to require that the pipe is empty and the pump is stopped when deconnecting a pipe from a pump. Otherwise, it is difficult to predict what happens to the pump and to the pipe (where is the water flowing?). If we accept only safe reconfigurations, we have:

Property 1 *The state of any component of a system is not changed by a reconfiguration.*

This safety property of a reconfiguration is important since one knows exactly what happens to a system when it is reconfigured. It allows consequently to extend the (decentralized) on-line diagnosis in order to take into account reconfiguration actions in an easy way as sketched in Section 6.

5 ILLUSTRATION

In this section, we illustrate the way our model react to the occurrence of reconfiguration actions and external events in a simulation way. Starting from the very beginning (a set of unconnected components), a sequence of interleaved (re)configuration actions and external events is simulated and the way these events are taken into account by the model is commented. We also show the modification of the global model in different topologies due to the reconfigurations.

5.1 One pipe delivering water

Let us suppose that we start from scratch. The components are still not connected and are in their initial states: the pump is OK and *off* (state 1) and the two pipes are OK and $empty$ (state 1). The model system is then $(Comp, Mod, Top)$, where $Comp = \{\text{P}, \text{PI1}, \text{PI2}\}$, Mod is such that the models of the pipe and the two pipes are instances of the models given in Figure 2 and Figure 3 and $Top = \emptyset$.

Let us suppose now that we want to deliver water to a container that is close to the pump. The operator has first to connect the pump to a pipe (we choose PI1) and to start the pump.

A first reconfiguration consists in connecting the pump and the pipe. $\mathcal{R}_1 = (DA_{\mathcal{R}_1} = \{\},$ $CA_{\mathcal{R}_1} = \{((\text{P}, p{::}out), (\text{PI1}, pi1{::}in), |)\})$. ($|$ is the synchronisation set presented in subsection 3.2, *i.e.*, the identity function.) No connection is removed and a connection between the output of the pump (P) and the input of the first pipe (PI1) is added. It can be checked that this reconfiguration is safe with respect to the current (initial) states of the components. After reconfiguration, the model of the system is described by $(Comp, Mod, Top')$, where $Top' = (Top \cup CA_{\mathcal{R}_1}) = CA_{\mathcal{R}_1}$.

The behavior of the system depends only on the behaviors of the pump and of the pipe. The synchronization is realized on the connections regarding the flow through these two components. The global model of the system, which depends only on the pump and the connected pipe, is given on Figure 4 (the internal messages are removed for simplification).

The state of the pump is 1 and the state of the pipe PI1 is 1. In order to start the pump, a first action is executed to put on the pump which corresponds to sending the message on on the p^{ext} point of the pump. The pump goes into state 2 sending the message h on its point called out. The topology indicates that this message is received by the input of the pipe as h. The pipe then changes its state into 3 and sends the message h to its output. Since the output of the pipe is not connected, the message is lost. In the explicit model of the connected system, this change corresponds to going from the state $(1, 1)$ into the state $(2, 3)$ by the transition labeled $\{(\text{P}{::}ext, on), (\text{PI1}{::}in, h)\}$ | $\{(\text{P}{::}out, h), (\text{PI1}{::}out, h)\}$.

Let us now consider the occurence of a fault on the pipe, which starts leaking. The state of the pipe is then changed to 6. Its outflow becomes low. If we consider the global model, the system evolves to the state (2,6) since the leaking of the pipe does not influence the state of the pump.

5.2 Two pipes delivering water

Let us now suppose that we want to provide water to another container, which is farther from the pump than the previous one. So, we decide to connect the second pipe to the first one.

Since the current state of the first pipe is 6, its output connection cannot be reconfigured (the set of states in which the pipe can support reconfiguration on its output connection point $G_{PI}(out)$ does not contain the state 6).

The pump has to be stopped. A message *off* is received on the external point of the pump, which reacts by sending the message z to

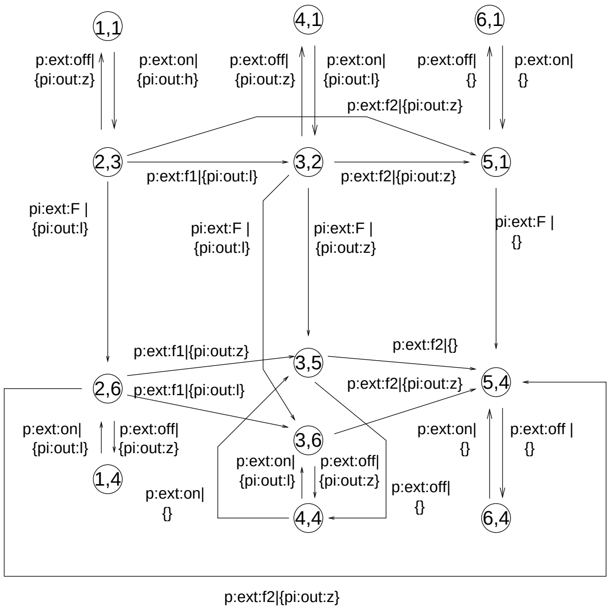

Figure 4. Model of the System

the pipe. The new state of the pipe is then 4 and from that state the reconfiguration is now possible.

The reconfiguration is $\mathcal{R}_2 = (DA_{\mathcal{R}_2} = \{\},$ $CA_{\mathcal{R}_2} = \{((\text{PI1}, pi1{::}out), (\text{PI2}, pi2{::}in), |)\})$. The resulting topology is: $Top'' = \mathcal{R}_2(Top') = \{((\text{P}, p{::}out)\ (\text{PI1}, pi1{::}in), |),$ $((\text{PI1}, pi1 :: out), (\text{PI2}, pi2 :: in), |)\})$. The explicit model of the system is too large to be given.

The pump may now be started again. A message on is sent on its external point. The pump delivers a high flow to the first pipe, which only delivers a low flow to the second pipe since it is leaking.

Let us now consider that a failure occurs over the pump. The message $f1$ is received by the external point of the pump. The pump sends a low flow to the first pipe. As the model of a leaking pipe is not deterministic as explained in Section 2, (two transitions exit from state 5 in the model of Figure 3), two output flows can be predicted at the output of the pipe.

This section illustrates how the system model takes into account the reconfiguration actions. Due to Property 1, the states of the system components are well-identified even when reconfiguration actions happen on-line. It is then possible to rely on it in a diagnosis perspective: observable events are now collected and diagnosis candidates are looked for by confronting them to those predicted by the model. Let us recall that all reconfiguration actions are supposed to be controlled by the operator and thus observable. In the next section, we explain how our decentralized diagnosis approach, developed for topology-stable systems [9] is currently extended to reconfigurable systems. This diagnosis step is now under development and will be the subject of a next paper.

6 FUTURE WORK: DIAGNOSING RECONFIGURABLE SYSTEMS

Several frameworks have been proposed for a decentralized approach to diagnosis of discrete-event systems [7, 6, 1, 9]. In our diagnosis decentralized approach [9], as in related ones [7, 6], each component is observed by sensors that send observations to a single supervisor. The contribution proposed by [9] consists in computing local diagnoses for subsystems and then to efficiently merge these local diagnoses to obtain a global diagnosis for the whole system. The main role of the merge operation is to filter the diagnoses which do not satisfy the synchronization constraints between components. To extend this approach to reconfigurable systems, it is sufficient (thanks to the hypotheses we take and to the Property 1) to correctly update the synchronization constraints when reconfiguration actions occur. These synchronization constraints are then used as before by the merge operation when computing the global diagnosis from the local diagnoses.

When incrementally computing the diagnosis as in [9], the observations are considered on successive *temporal windows*. The current diagnosis is updated by taking into account the flow of observations of the next temporal window. In the case of reconfigurable systems, the definition of these temporal windows, and especially the property of safety, has to be extended with respect to reconfiguration actions.

When dealing with on-line diagnosis, it is well-known that the efficiency of the algorithm is a major problem. The reason is that most of decentralized diagnosis approaches rely on explicit representation of models (often automata), which is prohibitive, even with partially compiled representations as diagnosers. As in our recent works, we intend to use techniques such as Partial Order Reduction [8] or Inversibility [4] to improve the computation of diagnosis by exploiting the structure of the model. In the same vein, symbolic representation or model-checking techniques [2], known to significantly improve search on automata, could be used for the diagnosis of potentially large reconfiguration systems.

Future work will consider whether some of the assumptions we made can be relaxed. A first case could introduce reconfigurations actions occurring at any time (for example an observation sent by a sensor before a reconfiguration action and received by the supervisor once the reconfiguration is performed). Another case could be to consider more complex reconfigurations as reconfiguration actions taking time or connections with delays or loss of observations.

7 CONCLUSIONS

On-line reconfiguration refers to the modification of the architecture of a system involving the creation, removal or replacement of elements while preserving the continuity of service.

In this paper we presented the foundations of a new approach to reconfigurable discrete-event systems modelling with a further diagnosis objective. We first proposed to extend the decentralized approach to reconfigurable systems in order to describe more precisely the connections between the components. The system model relies on the model of each component and on the topology model that describes the connections between components through connection points. We then introduced the reconfiguration formalism defined as a set of connection and deconnection actions. Since a crucial issue is to integrate on-line reconfiguration during the decentralized diagnosis task, we stated five hypotheses and an important property of reconfiguration called safety. Based on this property, the new state of the system can be inferred without ambiguity once a reconfiguration has been realized. Then, the proposed modelling of reconfigurable systems can be used with a few adjustments for on-line diagnosis of discrete-event systems.

Our current work consists in adapting the decentralized algorithm proposed by [9] to treat reconfiguration actions. The most important for future work has been given at the end of Section 6. The first item proposes to continue improving the efficiency of the algorithm by using symbolic representations and reduction techniques. The second item consists in relaxing our assumptions to diagnose more general reconfigurable sytems where some observations are delayed or lost and when reconfiguration actions take time. Finally, as a priority task we plan to experiment this approach on telecommunication networks as we did in [9].

References

[1] A. Aghasaryan, E. Fabre, A. Benveniste, R. Boubour, and C. Jard, 'Fault detection and diagnosis in distributed systems: an approach by partially stochastic Petri nets', *Discrete Event Dynamic Systems*, **8**(2), 203–231, (1998).

[2] B. Brard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and Ph. Schnoebelen, *Systems and Software Verification. Model-checking Techniques and Tools*, Springer, 2001.

[3] M.-O. Cordier and C. Dousson, 'Alarm driven monitoring based on chronicles', in *4th Symposium on Fault Detection Supervision and Safety for Technical Processes (Safeprocess'00)*, pp. 286–291, (2000).

[4] M.-O. Cordier, A. Grastien, C. Largouët, and Y. Pencolé, 'Efficient trajectories computing exploiting inversibility properties', in *14th International Workshop on Principles of Diagnosis (DX-03)*, pp. 93–98, Washington, USA, (2003).

[5] J. Crow and J. Rushby, 'Model-based reconfiguration: Toward an integration with diagnosis', in *National Conference on Artificial Intelligence*, pp. 836–841, (1991).

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

[7] G. Lamperti and M. Zanella, *Diagnosis of Active Systems*, Kluwer Academic Publishers, 2003.

[8] D. Peled, 'On model checking using representatives', in *5th International Conference on Computer-Aided Verification (CAV-1993)*, pp. 409–423, (1993).

[9] Y. Pencolé, M.-O. Cordier, and L. Rozé, 'Incremental decentralized diagnosis approach for the supervision of a telecommunication network', in *IEEE Conference on Decision and Control*, pp. 435–440, Las Vegas, USA, (2002).

[10] L. Rozé and M.-O. Cordier, 'Diagnosing discrete-event systems : extending the "diagnoser approach" to deal with telecommunication networks', *Journal on Discrete-Event Dynamic Systems: Theory and Applications (JDEDS)*, **ftp://ftp.inria.fr/INRIA/publication/publi-ps-gz/RR/RR-4957.ps.gz**, (2002).

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

[12] M. Stumptner and F. Wotawa, 'Reconfiguration using model-based diagnosis', in *Tenth International Workshop on Principles of Diagnosis (DX-99)*, pp. 266–271, (1999).