Hierarchical design of distributed Fault Tolerant Control systems

Marta Capiluppi and Andrea Paoli

Abstract — This work deals with the description of a design procedure for hierarchical Fault Tolerant Control (FTC) of large, distributed systems. Following a functional perspective, a procedure for the modular design of the diagnostic and reconfiguration algorithms which run at different levels of the hierarchy is presented. The whole procedure is applied to an hydraulic benchmark system.

I. INTRODUCTION

In large systems, every component provides a certain function and the overall system works satisfactorily only if all components provide the service they are designed for. For this reason fault tolerance (see [1]) is a key issue in distributed systems because a single component failure, propagating into the system structure, may lead to a catastrophic system failure. Anyway in the design of a distributed system it is always possible to implement well-defined error-containment regions. An exhaustive description of distributed systems can be found in [2], while, concerning the topic of fault-tolerance in distributed systems, a good introduction to the problem can be found in [3], [4], [5] and [6].

This work attempts to present a unified framework for fault tolerant control of distributed systems. Following a functional approach, a modular/hierarchical fault tolerant control architecture and some design guidelines. The hierarchical architecture for fault tolerant control of a distributed system is a three-level architecture: the lower level is the process level, the intermediate level is the control level and the upper level is the supervision level. According to the description given in [7], the process level is divided in partial processes (achieving sub-functionalities of the main functionality of the system) and physical resources. The implementation of each partial process needs allocation of resources, i.e. plant components and controller modules; this allocation is dynamic, dependent on reconfiguration decisions. The control level consists of control, monitoring and reconfiguration functions and is denoted as fault tolerant module level (a Fault Tolerant module stands for a possible hierarchical structure of modules). The supervision level aims to monitor performances of the system and (optimally) allocate processes into resources. The application of such a framework to a physical system is presented in the second part of the paper.

II. DESIGN OF THE ARCHITECTURE

According to the previous description a general criterion to design the distributed fault tolerant control system is the functional criterion. A tool by which it is possible to highlight the functionality map linked to each partial process is the Functionality Tree (see [8]). This is a graphical tool by which the global functionality of the system, the root of the tree, is expressed in terms of sub-functionalities (i.e. intermediate nodes of the tree) instrumental for the achievement of the main functionality. Strictly related to the functionality tree is the Fault Tree which shows the map of loss of functionalities as a consequence of faulty conditions (see [1], [8]). The fault tree can be interpreted as a complementary version of the functionality tree. Losses of functionalities at each level of the tree are seen as caused by losses of functionalities sitting at lower levels until the elementary cause, given by the physical fault, is reached. The problem of diagnosis is strictly connected to the ability of detecting a possible loss of functionality due to a fault. In this respect it makes sense to define failure modes as the loss of functionalities (nodes of the fault tree) which are detectable by static as well as dynamic elaboration of the available measures. In general it is possible to associate to each failure mode in the fault tree one or more residual signals. The joint elaboration of residual signals should indeed allow to detect a faulty condition and, in particular, to isolate the failure modes closer to the leaves of the tree. In this sense the functionalities hierarchy described in the functionality tree reflects in the hierarchy of the diagnostic system aimed to detect the losses of these functionalities. In particular a nested interpretation can be given to the overall residual matrix as it can be obtained by nesting different residual matrices linked to different levels of the fault tree. The graphical description given by the functionality and fault tree can be used to identify a hierarchy also in the reconfiguration algorithm: we define Reconfigurable Functionalities the nodes of the functionality tree on which the designer has some degree of controllability. It is possible to identify a set of control functions associated to each reconfigurable functionality, namely a set of reconfiguration actions. Linked to each reconfigurable functionality, there exists a local supervisor which has the role of choosing the reconfigurable functionality which is the closest to the estimated failure mode. As a matter of fact the rationale behind this criterion is to reconfigure the system as close as possible to the original cause of the fault condition, in order to shrink the region in which the deviation from nominal...
conditions is compacted. According to this description, each local reconfiguration supervisor can be thought as composed by three fundamental parts: an FDI unit, an Event Generator unit and a Decision Logic unit. The goal of the FDI unit is to provide, on the basis of the different local diagnostic information (residual signals), an estimation of the failure mode observed in a specific process/sub-process. The failure mode estimation is then processed by the Event Generator unit whose aim is to raise requests of working mode changes. All the requests of WM changes must be then managed by a decision logic unit which validates or not the requests according to a global vision of the system.

III. DESIGN OF THE SUPERVISORY SYSTEM

The decision logic is the reasoning engine of each supervisor. In the sequel a procedure to design the decision logic unit at each level of the hierarchy is briefly presented (for a more detailed description the reader is referred to [9] and [10]).

A. Low level supervision

The decision procedure for the decision logic unit of each local supervisor (intermediate level supervisory system) can be described by the following steps. The step number 0, detailed in the following, is indeed not needed for the supervisor design but only for verification purposes (see [11]).

- **Step 0:** Starting from the set of working modes associated to a local reconfigurable functionality it is possible to build the Discrete Event System (DES) modeling the system with respect to the considered fault. The initial state (let call it WMO) models the nominal condition for the system. A fault event \( f \), observable but not controllable, moves the system into a faulty state (F).
  
  Here it is possible to force one of the reconfiguration actions (working modes) enlightened in the tree as possible reconfiguration actions for the fault. Events named \( wm_i \) are controllable and observable events used by supervisors to force a reconfigured working mode (state WMJ). Repeating the procedure for all possible faults affecting the system we obtain a DES modeling the behavior of the system. In general this automaton (named \( G \)) will be composed by a nominal state, a set of faulty state and a set of reconfigured state.

- **Step 1:** Consider now the \( j \)-th reconfigurable functionality at level \( i \) in the functionality tree. According to the specifications which are behind the reconfiguration of the specific functionality, it is possible to design a DES composed by a number of states WM\( i \) whose activation is governed by events \( wm_i \). These ones describe the set of reconfiguration actions which must be taken in order to reconfigure the specific functionality and, possibly, functionalities at lower levels. It is worth noting that the events \( wm_i \) can be, in principle, both controllable and non-controllable by the supervisor\(^1\).

- **Step 2:** The DES designed in Step 1 is completed with an activating prefix given by concatenation of failure mode events \( (f) \) and activated events \( (req_k) \). The automaton modeling the \( i \)-th level specification will be denoted as \( H_i \). Activating event \( req_k \) can be used by event generator to set the \( i \)-th level reconfiguration.

- **Step 3:** The controlled behavior of the system at level \( i \) \( (i = 1 . . . n) \), denoted with \( K_i \), can be computed as \( K_i = H_1 \parallel . . . \parallel H_i \parallel G \). Note that it is not required \( H_i \) by itself to be controllable with respect to \( G \), but, if \( H_1 \parallel . . . \parallel H_n \) is controllable and observable with respect to \( G \), the controllability and observability theorem holds; hence there exists a unique modular controller given by the set of \( n \) supervisors \( H_1, H_2 . . . H_n \).

B. High level supervision

In case the reconfiguration of a certain partial process has impact in the resource allocation and/or the change of working mode in a certain module has to be joined to a change of working mode in a different module, then the responsibility of orchestrating the new working mode switching is demanded to the high level Group/Global supervisor. This one is composed just by a Decision Logic Unit which processes all the events generated by the different Local RRM and manages the working mode changes involving reallocations in resources linked to different groups. Moreover the state of the Group and Global RRM can change due to external commands issued by external operators. The starting point is to identify the specifications which are behind the design of the decision logic. These are precisely presented in the following:

- **Group Selection:** how the different partial processes and resource units can be grouped.
- **Modules/Resources map:** an off-line planning on how the different working modes associated to a specific module can be allocated in the available resources.
- **Partial processes as DES:** the outcome of the (modular) design phase regarding the Local reconfiguration manager is a DES describing the desired behavior of the particular partial process. This is a set of states associated to the different failure and reconfigured situations and events describing transitions between states.
- **Resource Units as DES:** this task amounts in describing local resources as Discrete Event Systems. In the simplest case the states describing the status of resources reduce to three: idle (namely the resource is capable to run additional functionalities), busy (no

\(^1\)Non-controllable events are, for instance, due to implicit reconfigurations which do not need an explicit activation (passive/implicit fault tolerant controllers).

\(^2\)With \( \parallel \) is denoted the classical parallel composition operation between automata. For further details see [11].
other functionalities can be located on that resources) and faulty (a resource monitor has detected a local fault, for instance of a computer). The busy state can in general be split in several sub-states expressing different cases in which the resource can be busy. The exogenous events can be divided in change of working modes inducing transitions between idle and busy states and occurred faults, i.e. non-controllable but observable events arising whenever the local resource monitor detects a fault in the supervised resource.

- **Reconfiguration specifications**: this task involves the specifications regarding interlaced reconfigurations between modules. The specification at this level regards the identification of possible conflicting Working Modes and consequent remedial actions.

- **Working modes Urgency**: this information is needed whenever a reconfiguration must be actuated in presence of limited resources.

The composition of the discrete models of the functional units and physical resources, yields an automaton which captures the whole information about the feasible working modes according to the actual working mode and to the resources availability. From this automaton, the decision logic can be designed following the supervision theory on the basis of performance specifications and reconfiguration requirements. The algorithm to design the Group Global supervisors can be described by the following steps:

1) Replica of the DES associated to a module according to the Modules/Resources map: whenever a particular partial process can be allocated onto different resources, it is necessary to model its image on the particular resource. Since a supervised partial process is described by a DES ($K_n$), this step can be carried out simply by building a replica of $K_n$ to model its image on the $j$-th resource (let call this new DES $K_{n/j}$).

2) Since a partial function can be allocated just on a subset of the possible physical resources, it is basic to model the possibility that the particular process is in standby on the considered resource. For this reason the automaton $K_{n/j}$ is enriched with a new state $SB_j$ and a controllable event called $sb_j$. If state $SB_j$ is active in $K_{n/j}$ then the $n$-th module is not running on the $j$-th resource. Event $sb_j$ is used by high level supervisors to move process from a resource to another.

3) The group selection information and the Modules/Resources map together tell which partial processes and which resources compose a group. Hence it is possible to build the $i$-th group DES model (named $GR_i$) as a composition of all the possible images of partial processes and all the resources.

4) Using the information coming from the Modules/Resources map, the Reconfiguration specifications and the working modes urgency it is possible to design a specification controllable with respect to $GR_i$ and, in this way, the $i$-th Group supervisor ($GR_i$).

5) Finally the controlled behavior of the $i$-th group can be found as $L_i = GR_i || GR_i$. The open loop behavior of the system is therefore obtained composing all the controlled group behaviors $L_i (L = \parallel_{i=1...r} L_i)$. Designing a controllable specification with respect to $L$ means to define the Global supervisor behavior.

IV. AN APPLICATION

We present in the following the application of the above described procedure to an hydraulic system.

A. System description

The two-tanks system (see [12]) is composed by two tanks supplied by two pumps with flow rates $Q_1$ and $Q_2$ (see fig.1). The two tanks are connected through two redundant pipes with valves $V_{12}$ and $V'_{12}$. The output flows of the two tanks are mixed through valves $V_{F1}$ and $V_{F2}$. The system is equipped with two level sensors ($L_1$ and $L_2$) measuring liquid heights in the tanks and five flow-rates sensors measuring flows $Q_1$, $Q_2$, $Q_{12}$, $Q_{F1}$ and $Q_{F2}$. The physical equations of the system are

$$S_1 \dot{L}_1 = - Q_{F1} - Q_{12} + Q_1$$

$$S_2 \dot{L}_2 = - Q_{F2} + Q_{12} + Q_2$$

$$Q_{12} = \text{sgn}(L_1 - L_2) R_{12} \sqrt{L_1 - L_2}$$

$$Q_{F1} = R_1 \sqrt{L_1} \quad (1)$$

$$Q_{F2} = R_2 \sqrt{L_2} \quad (2)$$

where $R_{12}$ is the throttling of valve $V_{12}$, $R_1$ is the throttling of valve $V_{F1}$, $R_2$ is the throttling of valve $V_{F2}$ and $S_1$.

$V_{12}$ is an electromechanical valve and its throttling $R_{12}$ is modelled as $k_1 V + k_2$, where $k_1$, $k_2$ are suitably sized parameters and $V$ is the applied electrical voltage.

$S_2$ are the sections of tanks 1 and 2 respectively. The mathematical model of the system is then

\[
\begin{align*}
S_1 \dot{L}_1 &= -R_1 \sqrt{L_1} - \text{sgn}(L_1 - L_2) R_{12} \sqrt{|L_1 - L_2|} + Q_1, \\
S_2 \dot{L}_2 &= -R_2 \sqrt{L_2} + \text{sgn}(L_1 - L_2) R_{12} \sqrt{|L_1 - L_2|} + Q_2.
\end{align*}
\]

(3)

Flows $Q_1$ and $Q_2$ are considered as control inputs of the system. The controlled outputs are

\[
\begin{align*}
y_1 &= Q_{F1} + Q_{F2} = R_1 \sqrt{L_1} + R_2 \sqrt{L_2}, \\
y_2 &= Q_{F1} = \frac{R_1}{R_2} \sqrt{L_1}.
\end{align*}
\]

(4)

These two outputs are required to follow two desired set points, denoted respectively as $y_1^*$ and $y_2^*$. Set points $y_1^*$ and $y_2^*$ can be rewritten as desired set points $L_1^*$ and $L_2^*$ for the measured levels $L_1$ and $L_2$:

\[
L_1^* = \left( \frac{y_1^* y_2^*}{R_1(1+y_2^*)} \right)^2, \quad L_2^* = \left( \frac{y_1^*}{R_2(1+y_2^*)} \right)^2.
\]

(5)

We consider two nominal working conditions for the system, assuming that the throttling of valve $V_{12}$ is constant: in the first one the valve is closed (decoupled system), in the other one the valve is open (coupled system). In both situations, using the flow-rates $Q_1$ and $Q_2$ as control variables, from (3) it is immediate to see that the system is controllable. In the first situation (decoupled tanks) model (3) becomes:

\[
\begin{align*}
S_1 \dot{L}_1 &= -R_1 \sqrt{L_1} + Q_1, \\
S_2 \dot{L}_2 &= -R_2 \sqrt{L_2} + Q_2.
\end{align*}
\]

(6)

It is possible to achieve control objectives (4) or (5) using two PI controllers. We call this first working mode WM0. In case of valve open, the model of the system is represented by (3). The system is therefore coupled and must be controlled in a coupled way in order to track again set points (5). To satisfy these objectives we can use an optimal control strategy. We call this second working mode WM0'.

To cover some significant detection and reconfiguration possibilities for this system four realistic faults are considered:

- **Actuator fault**: pump 2 can stuck to a constant value $Q_2 = Q_{20}$, namely it injects a constant flow in tank 2. This fault leads to a loss of controllability as it changes one of the structural constraints in (3).
- **Hardware fault**: valve $V_{12}$ can stuck to a constant value $R_{12} = R_{120}$ which can be 0 (namely the valve fails in stuck closed mode) or a constant finite positive value.
- **Leakage fault**: as leakage fault we consider the possibility of having a hole in tank 2, i.e. there is an undesired outgoing flow from tank 2; the dynamic of $L_2$ is then corrupted by a term $\delta Q_{F2}(L_2) = h \cdot L_2$ where $h$ is the section of the hole.
- **Level sensor fault**: the measure $L_{2m}$ of level $L_2$ can be corrupted by a constant bias $\delta L_2$, i.e. $L_{2m} = L_2 + \delta L_2$.

Note that all faults are supposed to occur in tank 2, anyway analogous faults can be considered for tank 1.

**B. FTC design**

Due to the considered faults a first functional analysis of the system leads to a first decomposition in a Fault Tolerant Control (FTC) module achieving fault tolerant control of tank 2 and a Fault Tolerant Measurement (FTM) module managing fault tolerant measure of liquid level (see [7] for details).

We now present some residuals generated mostly using structural considerations. A first residual signal is obtained through a test on the control loop of pump 2, in fact flow $Q_2$ from pump 2 is a controlled variable so it is a known internal state $Q_{2}^*$ of the controller. In case of steady state, $Q_{2}^*$ should be equal to the reference value for the output flow from pump 2 measured with a flow sensor:

\[
r_1(t) = Q_{2}^* - Q_{2m}.
\]

(7)

Signal $r_1$ is equal to zero in nominal conditions, while it is different from zero in case pump 2 is stuck or the measure $Q_{2m}$ is not correct. This means that residual signal $r_1$ is sensitive to fault $Q_{2}^*$. Due to structural properties of the system, constraints linking throttling of valve $V_{12}$ to other variables depend on the state of the valve itself. In case the valve is open we have the flow $Q_{12}$ between the two tanks. Thus, since $L_1$ and $Q_{12}$ are measurable, it is possible to estimate $L_2$ as

\[
\hat{L}_2 = L_1 - \left( \frac{|Q_{12}|}{R_{12}} \right)^2.
\]

(8)

and to generate a residual signal $r_3(t)$ sensitive to $\delta L_2$ as

\[
r_3(t) = |L_{2m} - \hat{L}_2|.
\]

(9)

On the other hand in case the valve is closed it is always possible to estimate $L_2$ by means of a non linear observer:

\[
S_2 \dot{\hat{L}}_2 = -R_2 \sqrt{\hat{L}_2} + \text{sgn}(L_{1m} - \hat{L}_2) R_{12} \cdot \sqrt{|L_{1m} - \hat{L}_2| + Q_{2m} + G(\hat{L}_2 - L_{2m})},
\]

where $G$ is a suitable tuned negative gain. The residual signal $r_2(t)$ sensitive to $\delta L_2$ is generated using the estimation error:

\[
r_2(t) = |\hat{L}_2 - L_{2m}|.
\]

(10)

It is possible to use $\hat{L}_2 - L_{2m}$ to reconstruct the sensor bias and thus to estimate the level value. Equation (2) states that in case of nominal conditions, the outgoing flow from tank 2 depends only on level $L_2$ and on throttling $R_2$ of the outgoing valve. In case of leakage in tank two this relation does not hold anymore, since there is another outgoing flow. This means that the relation becomes:

\[
\frac{R_2}{L_2} = \frac{Q_{F2} + \delta Q_{F2}}{L_{2m}}.
\]

For this reason signal

\[
r_4(t) = R_2 \sqrt{L_2} - Q_{F2}.
\]

(11)

is equal to zero in nominal conditions while it is different from zero in case of leakage in tank two or in case of a...
misreading of the level sensor: \( r_4(t) \) is a residual sensitive to faults \( \delta Q_{F2} \) and \( \delta L_2 \). Finally let assume that the valve \( V_{12} \) is monitored with an hardware electrical test (for example an electrical signal consistency test). This test will result in signal \( r_7(t) \) which is able to detect a fault on valve \( V_{12} \) (more specifically it is able to detect the situation in which the connection valve is stuck, namely fault \( \overline{V}_{12} \)). It is easy to verify that with this set of residual signals the system is fully detectable and isolable with respect to the set of faults considered (see [10]). Now we can consider some possible reconfigurations for the presented faults. First let consider the fault on pump 2: pump 2 is stuck to a constant value such that \( Q_2 = \overline{Q}_2 \). If valve \( V_{12} \) is closed, the system is represented by (6) (the two tanks are decoupled). In this case we lose a control variable \( (Q_2 = \overline{Q}_2 = \text{const.}) \), and hence one degree of freedom. Since it is not possible anymore to achieve both control objectives in (4), one of the two is chosen and the trajectory on \( L_1 \) is reconfigured in order to achieve this objective. Because the incoming flow in tank 2 is constant, the level in this tank will stabilize to a constant level \( \overline{L}_2 \) which can be measured. Hence if the sum functionality must be preserved then we compute the new trajectory \( L_1^* \) as \( L_1^* = \left( \frac{y_1^* - R_2 \sqrt{L_2}}{R_1} \right)^2 \) and we will call this working mode WM1. If the ratio functionality must be preserved then we compute the new trajectory \( L_1^* \) as \( L_1^* = \left( \frac{R_2 y_2^* \sqrt{L_2}}{R_1} \right)^2 \) and we will call this WM2. In both cases we do not need an estimation of \( \overline{Q}_2 \) because we use the measure of \( L_2 \). Note that in these two cases the control law does not need to be changed. If valve \( V_{12} \) is open, the system is represented by (3). We must again decide which of the two control objectives we want to satisfy, because we have only one control input. Suppose we want to track \( y_1^* \): we can use a PI controller on the error \( e = R_1 \sqrt{L_1} + R_2 \sqrt{L_2} - y_1^* \). Note that even the implementation of this control strategy does not require the estimation of \( \overline{Q}_2 \). We will call this working mode WM3. Similarly, by forcing objective \( y_2^* \) instead of \( y_1^* \) and following the same procedure as above, we can define a working mode WM4 in which the objective \( y_2^* \) is forced, without estimation of \( \overline{Q}_2 \), through a PI controller on the error \( e = R_3 \sqrt{L_1} + R_2 \sqrt{L_2} - y_2^* \). A leakage fault on tank 2 implies a parametric change in the model of the system and can be tolerated locally using a robust control law. In this sense it is possible to define a reconfigured zero impact working mode WM5 in which the local control law on \( Q_2 \) is robust to this fault. The sensor fault can be managed by the FTM which estimates the interested variable, hence no explicit reconfiguration is needed regarding fault \( \delta L_2 \), because we suppose to always have a reliable estimation \( \hat{L}_2 \) of the value of \( L_2 \). This is a zero impact reconfiguration called WM6. Consider now the fault on valve \( V_{12} \). If the valve is stuck close we can switch to valve \( V_{11}^* \) and use \( R_1^* \), due to the parallel hardware redundance. This working mode is denoted with WM8. All the previous considerations can be used to build a functionality/fault tree (see [8]). For the FTC module the fault tree is represented in figure 2. From this figure it comes out that when we cannot achieve both local objective \( L_1^* \) or \( L_2^* \), then global objectives \( y_1^* \) and \( y_2^* \) cannot be achieved at the same time. Achieving local objective \( L_2^* \) is possible only if both desired control input flow \( Q_1^* \) and expected output flow rate \( Q_{F2}^* \) are achieved. The output flow rate can be different from the expected one only if a leakage fault has occurred. The desired control input flow is obtained if the actuator works and if the control law is right. The only fault on the actuator considered is the stuck of the pump \( \overline{V}_{12} \). Fault on valve \( V_{12} \) is not considered here because managed by the resource monitor. For the FTC module the fault tree is represented in figure 3. From relation (2) it comes out that the main functionality (estimation of the level in tank 2) can be obtained through the measure \( L_{2}\text{m} \) of the level itself or through its computation using this analytical redundancy relation (called \( f(L_2) \) in figure 3). The measure can be affected by the sensor fault \( \delta L_2 \); the estimation through redundant relations can be affected by the leakage fault \( \delta L_k \). In figure 2 residual \( r_1 \) is associated to the loss of desired input flow, while residual \( r_4 \) is associated to the loss of expected output flow in the FTC fault tree. In the FTC fault tree of fig. 3, residuals \( r_2/r_3 \) are associated to the loss of measure, while residual \( r_4 \) is associated to the loss of estimate itself. As for the reconfigurable functionalities of the system, we stated above that fault \( \overline{Q}_2 \) implies forcing one of the two control objectives of the system. This is done at the higher level of the fault tree (the higher functionality is changed) as can be seen in fig. 2 from the label c. The leakage fault is compensated through a robust control and the labelled functionality in

```
Fig. 2. Fault tree for the FTC module.
```

---

V. Conclusion

In this work a design procedure for hierarchical Fault Tolerant Control (FTC) of large, distributed system has been presented. It has been shown how, under a functional perspective, it is possible to identify a procedure for the modular design of the diagnostic and reconfiguration algorithms which run at different levels of the hierarchy. Moreover it has been shown how to use the theoretical machinery of the supervisor theory of discrete event systems to design a hierarchical decision logic algorithm. An example of application of this design procedure is then presented.

REFERENCES

Fig. 4. Supervisor specification: first level LRM (a); top level LRM (b); LRM for FTM (c); GRRM (d).

Fig. 5. Objective $y_1$ (a) and $y_2$ (b) with fault on pump 2 at time $t = 2000$ sec and set-point changes. The controller is reconfigured forcing objective $y_1^*$ (WM3). Set-point values are represented by the straight line.