ADS2 : Anytime Distributed Supervision of Distributed Systems that Face Unreliable or Costly Communication

Cédric Herpson* and **Vincent Corruble** and **Amal El Fallah Seghrouchi**
Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France
CNRS, UMR 7606, LIP6, F-75005, Paris, France
e-mail: firstname.lastname@lip6.fr

Abstract

The purpose of a supervision system is to detect, identify and repair any fault that may occur in the system it supervises. Nowadays industrial process are mainly distributed, and their supervision systems are still centralized. Consequently, when communications are disrupted, it slows down or stops the supervision process. Increasing production rates make this subjection to the state of the communications no more acceptable. To allow the anytime supervision of such systems, we propose a distributed approch based on a multi-agent system where each supervision agent *autonomously* handles both diagnosis and repair on a given location. This degree of delegation, never considered in the literature nor in the industry outside of the theoretical framework, requires to overcome several difficulties : How can one agent autonomously make a diagnosis with dynamically arriving information ? How can several agents may coordinate and reach a consensus on a given diagnosis or repair with asynchronous communication ? Finally, how to allow a human to trust the decisions of such a system ? This paper develops our proposal allong these three axis and evaluates ADS2 using an industrial case-study. Experiments demonstrate the relevance of our approach with an overall reduction of the supervised system down-time of 34%.

1 Introduction

Supervision systems were initially monitoring tools whose role was limited to collect and display information for their interpretation and use by the human expert. Today, the advent of complex and physically distributed systems leads to a semantic shift from supervision *tools* to supervision *systems*. Indeed, as the complexity of systems increases, humans can no longer process the flow of information arriving at each instant. The need to minimize the down-time and to improve system effectiveness requires the delegation of some of the decision-making power of the human supervisor to the supervision system. This requirement has lead to the (re)birth of a research community around the notions of autonomic computing [1] and self-* systems [2]. Our work lies within this context.

Within the *Dem@tFactory*[1] project, our objective is thus to improve the supervision of an existing digitizing chain distributed over several sites (see Fig 1). Different faults – single or multiple – can occur and alter or prevent the processing of the documents (e.g. a scanner quits working, a disruption of the connection between different sites halts or corrupts a data transfer, an OCR software is poorly set and generates unexploitable results, etc.).

Figure 1: In red, the communication links between the main sites of the digitization chain of the Dem@tFactory project. In yellow, the links with the current (centralised) supervision system.

Centralized supervision systems are currently the most common in industry. However, they do not perform well in asynchronous contexts. Indeed, communication malfunctions between the supervision system and the geographically distributed regions of the supervised system delay the repair and do not allow to quickly return to normalcy, even though a number of malfunctions may have local predefined repair procedures available. The unbounded communication time between the supervision and the supervised system is the main reason for this problem.

To overcome this lack of robustness when facing unreliable communications and to reduce the supervised system down-time, we present in this article ADS2 : a multi-agent architecture where each supervision agent autonomously handles both diagnosis and repair on a given location. The proposed architecture is composed of three mechanisms: A *decision mechanism*, a *coordination and consistency recov-*

[1]Project of the French R&D initiative Cap Digital federating 4 industrials and 3 laboratories.

ery mechanism and an *intertwining mechanism*. The *decision mechanism* tackles the dynamicity of the information available to an agent in order to make a diagnosis. The *coordination and consistency mechanism* deals with the problem of reaching a consensus between several agents on a global diagnosis (or repair) in a context of asynchronous communications. Finally, the *intertwining mechanism* address the problem of the size of the search-space in a multiple-faults context.

In this article, we first present our fault and repair model and the various assumptions made in section 2. We then describe the three mechanisms of our multi-agent architecture in section 3 to 5. We then demonstrate the viability of our proposal with experiments in section 6. Finally, we discuss related work in section 7 before concluding.

2 A Multi-Agent Architecture for the Supervision of Distributed Systems

Our architecture comes within the scope of fault-based model[2] approaches with spatially distributed knowledge. The supervision process is distributed among several autonomous agents having each a local view of the system to be supervised, and endowed with diagnosis and repair capabilities. The supervised system is partitioned into regions, each one is supervised by one agent. As illustrated in Fig. 2, the supervision agents (A_i) exchange information in order to establish a diagnosis and a repair consistent with the observations (O_j) they get from the various units of the supervised system (U_k). The links between the square units represent the standard workflow of the supervised system. The dashed arrows represent the fact that some elements may be reprocessed if the quality is not sufficient. The arrows between the units and the agents represent the communication links used to transmit alarms logs. The remaining links represent the communications between the supervision agents.

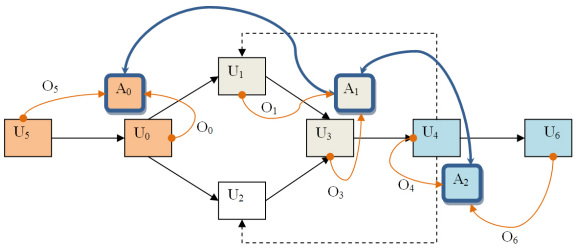


Figure 2: Example of our supervision systems deployed on a workflow.

2.1 Assumptions

We consider that communications are asynchronous and that there is no upper bound on transmission delay. We assume that the messages exchanged between supervised units may be lost or corrupted, and that some units are not supervised (*e.g.* unit U_2 on Fig. 2). This assumption is based on the fact that a complex industrial process commonly involves different actors that do not share their supervision information[3]. Moreover, we assume that the observations and the messages between agents can be lost but not corrupted. The agents are supposed to be reliable (no Byzantine behaviour). Finally, we consider that the simultaneous occurrence of different faults does not result in phenomena of masking observables.

[2]No model of the system's correct behaviour is available. The system can only use faults model, *a priori* known or dynamically learned from the system observation.

[3]subcontractors in the case of the Dem@tFactory project.

2.2 Fault model and repair plan

Let F be the set of known faults of a system S and R be the set of existing repair plans. The signature of a fault f is a sequence of observable events generated by the occurrence of f. The set of signatures of a given fault f is $Sig(f)$.

To be able to represent any temporal dependencies, each fault is modeled as a t-temporised Petri net (Fig. 3). Each fault is supposed to be repairable, that is to say that there exists at least one partially ordered sequence of atomic repairs r_k that repairs it (a repair plan).

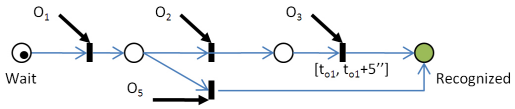


Figure 3: Let f be a fault that possesses 2 signatures. $Sig(f) = \{o_1o_5; o_1[o_2, o_3]^{[t_{o_1}, t_{o_1}+5'']}\}$. The o_i are the events observed on the supervised system. The t_{o_i} indicate the temporal constraints. Thus, $[t_{o_1}, t_{o_1}+5'']$ constrains the sequence of observations $[o_2, o_3]$ to appear under the 5 seconds that follow the occurrence of o_1 for f to be recognized.

The supervised system is partitioned into regions rg_j. Each supervision agent is associated with one unique region and knows the models of the faults that may occur in the region it oversees. However, a fault can cover several regions. In that case, an agent only knows the part of the model that concerns its region. Its model is completed with the names of the agents responsible for the others regions. This hypothesis allow to model workflow involving different actors that do not share their data.

$$Sig(f) = \{o_1^{rg_b} o_2^{rg_b} o_3^{rg_c} o_4^{rg_c}\} \Longrightarrow \begin{cases} Sig_{A^{rg_b}}(f) = o_1 o_2 A^{rg_c} \\ Sig_{A^{rg_c}}(f) = A^{rg_b} o_3 o_4 \end{cases}$$

Beside getting the models of faults, the issue of defining a global precedence relation between events that occur within the supervised system remains. Indeed, there is no common clock to the different regions. It is therefore necessary to add in each agent a stamping mechanism allowing to recreate this order relation. We will not detail here the concept of distributed clock.We consider in the following that the agents are able to recreate this partial-order relation.

2.3 Diagnosis and multiple faults

During the period of time $[t - \Delta_t, t]$, agent A_i collects a sequence of observations $seqObs_{A_i}(t, \Delta_t)$ generated by the occurrence of faults on the system. However agent A_i does not know which faults have occurred. It thus analyses $seqObs$ in order to determine the set of all faults $fp_{A_i}(t, \Delta_t)$ whose signatures partially or totally match elements of $seqObs$. A diagnosis dg is a set of faults that can explain $seqObs$. Dg is the set of all possible diagnoses of $seqObs$.

2.4 Fault cost and repair cost

Finally, each fault f (respectively each repair plan $rp(f)$) is associated with a cost of malfunction which depends of

the fault duration $Ct_{dysf}(f,t)$ (resp a cost of execution $Ct_{Ex}(rp(f))$). The cost of a diagnosis dg for the supervision system is the result of the aggregation of the respective costs of the faults that compose it. In the general case:

$$Ct_{dysf}(dg_i,t) = Aggreg_{f_j \in dg_i}(Ct_{dysf}(f_j,t)) \quad (1)$$

Similarly, the execution cost of a repair plan rp associated to a given diagnosis depends on the aggregation of the respective costs of the repairs that compose it. Thus, in the case the repair plan depends directly on the faults:

$$Ct_{Ex}(rp(dg_i)) = Aggreg'_{f_j \in dg_i}(Ct_{Ex}(rp(f_j))) \quad (2)$$

3 Agent Decision Model

We consider highly dynamic systems. Consequently, information available to an agent at a given time can be insufficient to determine with certainty which action to select. A supervision agent has thus to determine the optimal decision (D_{opt}) between the immediate triggering of the plan made under uncertainty ($Dimm_{opt}$), and a delayed action ($Ddelay_{opt}$) which lets him to wait and communicate with other supervisor agents during k time steps. This waiting time can yield information that reduces uncertainty and thereby improve decision-making. The counterpart is that the elapsed time may have a significant negative impact on the system. The *expected* potential gain in terms of accuracy must be balanced with the risks taken.

Let $Ct(x)$ the cost of an action x and $Ct_{wait}(k)$ the cost related to the extra time k before selecting a repair plan. The decision-making process of each supervision agent works as follows :

1. Observation gathering
2. Computation of the different sets of faults that can explain the current observations : Dg (set of diagnosis)
3. Determination of the immediate repair $Dimm_{opt}$ based on available information and on the constraints we chose to focus on (Most Probable Explanation, Law of parsimony, Worst case,...) and computation of its estimated cost $Ct(Dimm_{opt})$
4. A time t, an agent knows the set of the faults that may be occurring in the region it supervises $fp_{A_i}(t,\Delta_t)$. Knowing theirs signatures the agent is able to predict, for each fault of $fp_{A_i}(t,\Delta_t)$, the set of observables that can be expected to appear during the time interval $[t,t+k]$, with k an *a priori* fixed parameter. The agent uses these information to compute the waiting cost $Ct_{wait}(k)$, the expected potential gain of a delayed repair $Ddelay_{opt}$ and its associated cost $Ct(Ddelay_{opt})$.
5. Choice between the immediate repair $Dimm_{opt}$ and the delayed repair $Ddelay_{opt}$

This algorithm is executed at each time-step and by each agent when faults occur. The value k represents an upper bound delay as an agents' decision is updated each time an observation is received. We will detail in the following subsections the steps 3 and 4 relative to the determination of the immediate and delayed repair and of their respectives costs.

3.1 Immediate repair $Dimm_{opt}$

The knowledge of the different signatures of faults allows us to establish a list of potential diagnoses Dg. We sort these explanations according to available information and to the constraints we chose to focus on (e.g the most probable explanation). After this step , the first element of Dg is the diagnosis considered as the most relevant at the current time. It is then necessary to estimate its cost.

The cost of the immediate repair $Ct(Drep_{opt})$ must take into account the execution cost of the repair plan associated to the diagnosis retained (Ct_{Ex}, equation 2), as well as a cost representative of the potential error relative to this decision, Ct_{Err}. Indeed,if the only cost considered is the one of the execution of the repair plan, the final decision (step 5) will always favour an immediate action compared with a delayed one due to the additional waiting cost of the delayed action.

$$Ct(rp(dg_i)) = Ct_{Ex}(rp(dg_i)) + Ct_{Err}(dg_i|Dg\backslash\{dg_i\}) \quad (3)$$

The computation of the error cost Ct_{Err} relies on the fact that we assume that the good diagnosis – and so the good repair – belongs to the sorted list Dg of the potential diagnoses. Thus, in case of misdiagnosis when selecting the first diagnosis dg_1 of Dg, the system will lose a time equal to the execution time of the first repair plan ($Ct_{ExecTime}(rp(dg_1))$) which will be supplemented by the execution cost of the newly chosen repair plan ($Ct_{Ex}(rp(dg_2))$) associated to the 2^{nd} diagnosis of Dg. As this second choice may also turn out to be an error, we define Ct_{Err} recursively on Dg. Thus:

$$\begin{cases} Ct_{Err}(dg_1|[]) = 0 \text{// Dg is empty, the diagnosis is correct.} \\ \\ Ct_{Err}(dg_1|Dg\backslash\{dg_1\}) = P(dg_1|Dg\backslash\{dg_1\})\times \\ \left[\begin{array}{ll} & Ct_{ExecTime}(rp(dg_1)) + Ct_{Ex}(rp(dg_2)) \\ + & Ct_{Err}(dg_2|Dg\backslash\{dg_1,dg_2\}) \end{array} \right] \end{cases}$$

with $P(dg_1|Dg\backslash\{dg_1\})$, the probability that choosing dg_1 as the final diagnosis is an error.

3.2 Delayed repair $Ddelay_{opt}$

A time t, an agent knows the set of the faults that may be occurring in the region it supervises $fp_{A_i}(t,\Delta_t)$. The different faults models are represented using t-temporised Petri-nets (Fig. 3 page 2). The agent is thus able to predict, for each fault of $fp_{A_i}(t,\Delta_t)$, the set of observables that can be expected to appear during the time interval $[t,t+k]$, with k an *a priori* fixed parameter. Note that the agent uses the current transmission duration (computed over the interval $[t-\Delta_t,t]$) to determine the set of potential observations.

From this information, the agent builds the tree representing the set of all possibles futures working towards the current time plus k units of time, $Arb^{possibles}_{A_i}(k)$. Each node of the tree is associated with a set of observations and represent one possible future (Fig. 4 below). The agent then computes, for each node of the tree, the set of diagnoses that explain this future (Dg').

The agent can then compute, for each possible future, the immediate decision considered as optimal. At time t, the determination of the delayed decision with horizon k ($Ddelay_{opt}$) involves choosing between the various possibles situations. This choice is realised by sorting the first elements of each Dg' of the tree of the possibles futures with each other using the same criterion than the

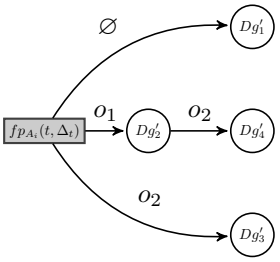


Figure 4: Illustrative example of a tree of the possibles futures.

one used to identify the immediate repair in the sub-section 3.1.

Once the delayed decision is identified, its cost $Ct(Ddelay_{opt})$ is established using Equation (3). We then have to add to this cost the waiting cost Ct_{wait}. This waiting cost represents the consequences of the faults on the supervised system during the time where no action was triggered. The computation of the waiting cost depends on the respective costs of the malfunctions associated to the remaining diagnoses and of the elapsed time.

$$Ct_{wait}(k) = Aggreg_{dg_i \in Dg}(Ct_{dysf}(dg_i, k))) \quad (4)$$

4 Distributed Supervision and System Consistency

In the previous section, we addressed the problem of one agent making a decision. However, as each agent has a local view of the system, a decision about a diagnosis and/or a repair frequently requires information and knowledge from other supervision agents. It therefore becomes necessary to reach a consensus on the decision to make.
However, distributed supervision works in a context of asynchronous and unbounded communication. Within these constraints the theorem of Fisher-Lynch-Paterson [3] states the impossibility of guaranteeing the achievement of a consensus between different components.

To circumvent this difficulty, the literature on supervision frequently introduce hypotheses on the quality of the communications. As our work tends to work under real-life hypotheses, we do not make any regarding the (un)reliability of the communication. We discuss in this section the use the multi-Paxos algorithm [4] to reach a consensus when the state of the communication allows it, and propose a consistency mechanism to restore a common view of the system by the agents after a unilateral decision taken by some of them.

4.1 Consensus algorithm

In the general case, establishing a consensus must meet the following properties : (Agreement) All correct processes decide the same value. (Integrity) Every process decides at most once. (Validity) Each value determined belongs to the set of proposed values. (Termination) Every correct process eventually decides in a finite time. However, in the context of Fisher-Lynch-Paterson theorem, the supervision system can only offer a guarantee of "best-effort". i.e, to assure that the consensus can be reached, but only if the system is stable on a sufficiently important period of time [5].

The multi-Paxos algorithm, initially developped for reaching an agreement in a network of unreliable processors, falls into this category. The interesting aspect of this algorithm is that it was designed to resist to halt failures - with recovery possibility - of a number of processes, including the coordinator. Its very low number of assumptions makes it operational in an environment with unreliable communications. These properties make it particularly suited to multiagent systems. Using the multi-Paxos, each agent is able to initiate, integrate or leave a coalition.

The fact that there is no upper bound on the time needed to reach a consensus will inexorably lead to some unilateral decision-making by agents or agent groups in case of communication disruption. This feature of our system guarantees the avoidance of deadlock situations when communications are too unstable to let the agents reach a consensus. However, this ability requires the introduction of an algorithm to restore a consistent view of the system state by all agents.

4.2 System consistency

Algorithm 1 works in the manner of producer-consumer with the decision-making process introduced in section 3. The two algorithms share, within an agent, a common inconsistency queue F_{inc}. When a coalition is left by at least one agent before reaching a consensus (due to a communication breakdown or to an agent's decision), the members of the coalition store their respective decision-making context (the current sequence of observables, the set of considered explanations and the list of agents which belong to the coalition) into their own potential inconsistency queue F_{inc}.

The consistency maintenance algorithm is available within each agent as a behaviour, it continuously observes the state of the queue F_{inc}. When an entry is added to F_{inc}, the algorithm is automatically triggered.

Algorithm 1 Check consistency

Require: Pattern observer on F_{inc}
1: **if** $F_{inc} \neq \varnothing$ **then**
2: Try to contact $F_{inc}.getFirst().getCoalition()$
3: **if** contact successful **then**
4: Send $F_{inc}.getFirst()$
5: Receive other agents decision context
6: Make pairing between local decision context and others.
7: **if** pairing is ok **then**
8: $F_{inc}.removeFirst()$
9: **else**
10: start new paxos instance
11: **end if**
12: **end if**
13: **end if**

This algorithm lets each agent find a match between its actions and those selected by the other members of the coalition. Thus, in case of faults due to past inconsistency decisions taken by the agents, they are able to trigger a sequential diagnosis and to discriminate initials disturbances from the consequences of their decisions.

When the potential inconsistency queue of an agent contains one element, the agent tries to resolve it. The agent tries to contact each of the agents of the coalition concerned with this potential inconsistency P_{inc}. If these agents are able to communicate (the communications are restored),

they will exchange their respective decision-making contexts. By comparing them, they will be able to determine whether the decisions made locally by the different groups of agents are consistent with one another. If this is the case (the faults are repaired, the system is in a stable state and correct), each agent removes P_{inc} of the queue. Otherwise, the subset of agents involved initiates a new coalition in order to resynchronize their respective views of the system and make a decision consistent at the system's scale. If communications are too unstable (or too costly), this consensus will not be reached, which results in adding a new entry in F_{inc}.

Restoring the consistency of the system state as it is perceived by the supervision agents is again relying on the stability of the communication links for a sufficient amount of time.

5 Intertwining Diagnosis and Repair Stages

In the previous sections, we endowed the supervision agents with decision-making and coordination mechanisms. These abilities allow the agents to dynamically adapt theirs behaviours to the current state of the communications and of the supervised system. In case of uncertainty regarding the decision to make, the agents are thus able to explore the solution space, collectively as well as individually. However, the large size of this set remains a problem. Indeed, it is both a source of misdiagnosis in case of local decision-making and the cause of a large number of supervision messages when a consensus must be reached. In order to reduce the complexity of the decision process, we adress in this section the question of obtaining the minimal set of diagnoses and of associated repair plans. To this aim, we discuss the idea of intertwining the diagnosis and repair stages.

This idea has been introduced by Cordier *et al* [6] on the formalization of self-healing systems [7]. Several failures may indeed have the same signature without calling into question the repairability of the system, all that is needed is that a repair be common to all of the faults involved (notion of *macrofault*).

However, restricted to the single-fault context, this formal model defines the diagnosability and repairability of a system as static properties that can be computed offline. This is not the case in the multiple-faults context. Indeed, the appearance of faults can prevent the triggering of a repair associated to another fault currently occurring in the system, and the possible situations are endless. Being able to represent this kind of interference is essential to our work. This led us to introduce context-dependent notions of diagnosability and repairability.

Definition 1 (Conditionnal Diagnosability)**.**

$$\begin{aligned} Diagnosable(f_i, t) &\iff \forall x \in C^D(f_i), x \notin ss(t) \\ &\iff C_t^D(f_i) = \varnothing \end{aligned}$$

A fault f is diagnosable at time t if none of the faults that may prevent its diagnosability (e.g if they share the same signature) is appearing in the system at this instant. This set of faults is the conflict set in diagnosis of the fault f (denoted by $C_t^D(f)$). Following the same reasoning, we can define $C_t^R(f)$ as the conflict set in repair of the fault f.

Finally, the uncertainty regarding the faults that are currently occurring on the supervised system, may conduct the supervision agents' to "believe" the occurrence of faults which are not real and which prevent the repair of the system. We call these situations virtual deadlocks. To disambiguate these situations, we added a relationship of innocuousness I to these definitions. Thus, for a fault f, a repair plan $r(f)$, its repair conflict set $C^R(f)$ and considering the current state of the system, I returns the set of sets of faults belonging to C_t^R on which the execution of repair plan $r(f)$ leaves the system unchanged. The result of this innocuousness relation is the set of disambiguation under repair, denoted by $D^R(f)$. Taking into account all this information, we are then able to propose an algorithm to plan the order of the repairs and to resolve some conflicts. We illustrate how it operates below :

Example : Let $F = \{f_1, f_2, f_3\}$ with $Sig(f_1) = Sig(f_2) = \{a\}$ and $Sig(f_3) = \{b\}$. $Rp(f_1) = \{r_1\}$, $Rp(f_2) = \{r_2\}$ and $Rp(f_3) = \{r_3\}$. Moreover, we know that $C^R(f_1) = \{\{f_3\}\}$ and that $D^R(f_2) = \{\{\overline{f_2}\}, \{f_1\}\}$. As the signatures of f_1 and f_2 are identical, it follows that $C^D(f_1) = \{\{f_2\}\}$ and $C^D(f_2) = \{\{f_1\}\}$. We assume that an agent detects the observables $a \wedge b$.

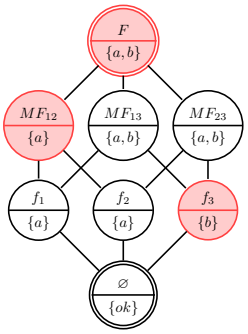


Figure 5: Diagnosis state for the agent : $dg_1 = \{f_2, f_3\}$, $dg_2 = \{f_1, f_3\}$

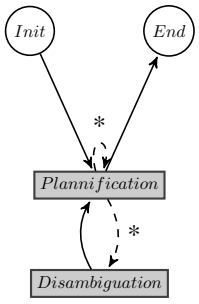


Figure 6: Operation scheme of the active repair algorithm

In Fig. 6, the initialization of the algorithm determines for each potential fault f_i all repair conflicts existing at the current time $C_t^R(f_i)$. The planning phase recursively builds the repairing order from C_t^D and C_t^R adding the faults whose conflict sets are empty, and then updates the remaining ones. At the end of this phase, if some faults remain, they potentially are in a deadlock. In our example, the agent has to choose between $\{f_2, f_3\}$ and $\{f_1, f_3\}$. As highlighted in Fig.5, the agent can repair f_3 but is unable to make a distinction between f_1 and f_2 (we assume that this conflict is virtual and that only one of these faults is occurring).

The disambiguation phase then attempts - from the proven innocuousness of some repairs in the current context - to solve these conflicts. If one of them is solved, the planning phase is retried after updating the conflict sets. If the disambiguation does not work, it means that the agent does not have, at the current time, enough information to solve the problem. The decision-making process previously introduced in section 3 is then triggered. In our example, repair r_2 is selected. If the system returns to normalcy, then both diagnosis and repair phases end. If not, the previous action guarantee the occurrence of f_1, and the associated repair plan is executed.

6 Experimental Evaluation

To evaluate our approach we developed a simulator for distributed systems. Based on the JADE multi-agent platform [8], our environment allows us to model both physical units and communications links, and to simulate the occurrence of failures in it. For a given simulation, a list of faults is associated with each site and each communication link (A communication link can increase its transmission time, a unit may stop working properly,...). Each fault is associated with one or more trigger conditions: a date and/or the occurrence of another failure. This allows us to simulate cascading faults. Our supervision system is deployed on this simulator. When running the simulation, some faults trigger the sending of an alarm message to the agent responsible for the site where they appear. The agents will try to determine the appropriate behaviour from these messages.

Our agent decisional model is generic. It can be instantiated with various criteria (the most probable explanation, the worst case hypothesis, etc.) depending on available information and on the constraints we choose to focus on.In the *Dem@tFactory* project, it appeared essential to consider the utility of a decision based on the cost of the occurrence of a given set of faults rather than on its occurrence probability. This reasoning led us to favor a robust decision criterion. The decisions taken by the agents will therefore rely on the worst case hypothesis.

The upper bound k which is the horizon considered by the agents of $ADS2$ for the computation of the delayed decision is set to 15 units of time. Moreover, we assume in these experiments that the respective costs of the faults that compose a diagnosis are additive. Finally, in order to have benchmarks for the evaluation of the principles underlying our architecture ($ADS2$), we also implemented a centralized supervision systems (SC) where all observables are transmitted to a single supervision agent.

6.1 Experimental setup

Our goal is to study the behaviour of the supervision system facing an industrial case-study.

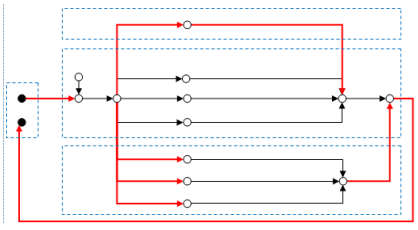


Figure 7: Workflow of the digitizing chain of the Dem@tfactory project.

Fig. 7 represents the digitizing chain of the *Dem@tFactory* used for the experiments. Each dotted rectangle corresponds to a factory situated on a given geographical location (2 in France, 1 in Madagascar and 1 in Mauritius). The circles correspond to the various process required to the digitization. The inter-rectangle links correspond to communications between the different factories, and the intra-rectangle one to local communications. All theses components are modeled in the simulator and can engender the occurrence of faults.

We fixed *a priori* the locations and responsibilities (the regions) of the supervision agents according to the geographical location of the units that compose the supervised process. We used a dataset provided by our industrial partners (8 GB of data corresponding to 48 hours of logs) to extract nineteen different faults models. From these data, we determined that the probability of occurrence of n faults per unit of time follows a Poisson distribution with parameter $\lambda = 0.043$. Finally, using information gathered from our partners, we were able to estimate the costs of the faults over time (constant, logarithmic,...) and the associated repair costs.

6.2 Experiments

Our experiments study the behaviour of the supervisions systems when varying the (heterogeneous) transmission cost. We used a random generator to affect a transmission time (between 0 to 30 units of time) to each transmission link for each time unit of the simulation. We arbitrarily set at 10% the probability of a link to get a transmission time greater than 1. In order to obtain a baseline, we first performed different simulations with homogeneous transmission costs.

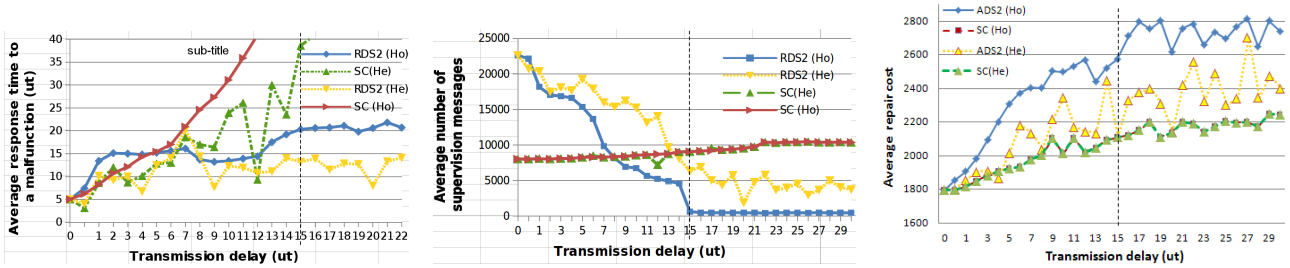
The performance evaluation is based on three criteria: (1) The average response time to a malfunction. (2) The average number of supervision messages exchanged. (3) The average total cost of repairs made during the experiment.

Figs. 8(a) and 8(b) present the evolution of the behaviour of supervision systems $ADS2$ and SC for the two first criteria in the case of homogeneous (Ho) and heterogeneous (He) communication links. The vertical bar at t=15ut is the horizon considered by the agents of $ADS2$ for the computation of the delayed decision.

Fig. 8(a) shows that our architecture is very robust, allowing the supervised system to rapidly recover from failures. The response time of $ADS2$ (Ho and He) progressively stabilized around 15ut, when the response time of SC increases over time and becomes higher than $ADS2$.This is due to the fact that the agents of $ADS2$ can decide to act without waiting for the reception of all the messages that come from the units of the supervised system.

We can observe an increase of the average response time of $ADS2(Ho)$ when the transmission delay is close to 15 ut. This is due to the parameter k of our algorithm, *a priori* fixed to 15 ut. This parameter defines the agent's horizon for the computation of the delayed decision. When the transmission time becomes greater or equal to k, an agent no longer sees interest in waiting or trying to exchange information with other agents of ADS2; so it decides to act despite the risk of making a mistake. The impact of parameter k is less important on $ADS2(He)$. Indeed, as the communication links are in this case heterogeneous, a supervision agent is still able to exchange information with some of the other agents. This leads to a better response time for $ADS2(He)$ than for $ADS2(Ho)$. This behaviour is clearly highlighted in figure 8(b). The number of messages exchanged by the agents of $ADS2$ drop with the increase of the transmission delay. We see a sudden drop of this number when the transmission delay becomes greater than 15 ut for $ADS2(Ho)$, confirming the local decisions-making of the agents.

Fig. 8(c) shows that the decisions of $ADS2(He)$ agents generate a limited repair extra-cost in comparison to the cen-



(a) Response time to a malfunction (b) Communication cost (c) Total cost of repairs

Figure 8: Experimental results. The curves corresponding to homogeneous/heterogeneous communication links are respectively marked with (Ho) and (He). The x-axis is the transmission delay. The y-axis correspond for each figure to one of the evaluation criteria.

tralised approach (9%). With the Dem@tFactory fault models, the overall gain regarding the supervised system downtime reach 34%.

Considering the reactivity of $ADS2$ and the limited repair extra-cost it generates, the communication extra-cost for low transmission delays can be considered as an acceptable consequences compared with a total-absence of supervision (SC).

Our next set of experiments evaluates the impact of the intertwining of the diagnosis and repair phases on the performances of the supervision system. In order to evaluate the impact of this behaviour for a supervision agent, we initially activated this capability in only one agent of ADS2. We realised 100 simulations. The 10 first simulations are performed with a number a simultaneous faults restricted to 1. Then the number is gradually incremented every 10 simulations to reach 10 simultaneous faults. For each simulation, the number of potential diagnoses considered by the agent is saved at 5 specific time steps. In order to obtain a baseline, we performed the same 100 simulation with the intertwining behaviour deactivated. Fig. 9(a) shows that the interleaving of the diagnosis and repair process does lead to a reduction of the diagnosis search space of an agent between 10 to 20% for the set of faults of the *Dem@tFactory* project.

Fig. 9(b) shows that the reduction of the number of potential explanations of each agent is of an extend sufficient to allow the agent to reduce the number of supervision messages. The everage response time to a malfunction is not significantly improved (Fig. 9(c)) but the repair extra-cost fall from +9% ($ADS2$) to 7.2% ($ADS2+$) (with p<0.05).

7 Related Work

The supervision of a system consists of four steps: Detection, Isolation, Identification and Repair. The literature aggregates the 3 first steps under the name FDI (Fault Detection and Isolation) [9]. Although several approaches for the distributed supervision of distributed systems have been proposed in the literature, whether work is from the diagnosis and control communities[10; 11] or from the multi-agent domain [12; 13], they do not cover the repair phase.

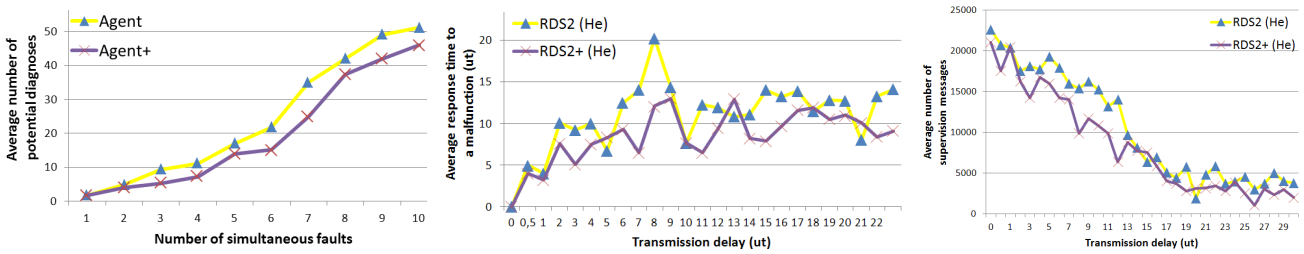
In the work from areas related to distributed systems, emphasis is placed on the distribution of available knowledge on the status and behaviour of the supervised system. Frohlich *et al* [14] and Roos *et al* [13] have addressed the question of the ability of a set of agents to determine an overall diagnosis according to the shape of this distribution. They have shown that to obtain a minimum overall diagnosis is NP-hard in the case of spatially distributed information and that the complexity of obtaining the diagnosis is independent of the communications costs engendered during its establishment[13].

Given these theoretical results, reducing the space of potential solutions is generally based on a hierarchical structure of the diagnosis agents [12] and on the choice of not returning to previously excluded explanation. Though the no back-track of past decisions guarantees convergence and termination of the algorithm, it is a source of diagnosis errors in an asynchronous environment. The *best-effort* approach we chose allows us to reduce these diagnosis errors, and the termination of the diagnosis algorithm is guaranteed through the anytime decision-making process of our agents.

To our knowledge, the work of Nejdl *et al* [15] is the only one that addresses the distribution of both the diagnosis and repair phases. However, placed at a relatively abstract level of analysis, this work makes the assumptions that communication links are reliable and that messages can be exchanged between agents at no cost. In a real situation these hypotheses are too restrictive. Indeed, to not consider the communication state may render the supervision system ineffective or inoperable. Our proposal does not make such assumptions.

The problem of online decision-making under uncertainty is the central point of the work by Horvitz [16], Hansen et Zilberstein [17] on the control of anytime algorithms. Indeed, they propose a formal framework to dynamically determine the time to stop a calculation taking into account the quality of the current solution and the cost of the algorithm computation.

The first distinction between these work and ours is that in their work the authors determine when to stop the computation based on the distance between the current solution and the optimal one. This requires knowing the optimal solution (or an estimatation) and to be able to dynamically determine this distance. In our work, talking about the quality of a solution (i.e a diagnosis) is meaningless insofar as a diagnosis is right or wrong, and its "value" is only known a posteriori. The second point of divergence is that we try to select a candidate (a diagnosis or a repair) among a set of potential solutions. The complexity of the task is therefore increased.



(a) Size of the potential diagnosis set (b) Response time to a malfunction (c) Communication cost

Figure 9: Simulation results when integrating the interleaving of the diagnosis and repair steps. The symbol "+" is associated to the components which integrate this additional mechanism.

8 Conclusions

We presented the first anytime multi-agent architecture for the supervision of distributed systems that is able to dynamically adapt its behaviour to the current state of the supervised system. In particular, the decision model allows each supervision agent to find a balance between a quick local diagnosis and repair under uncertainty, and a delayed, systemic one, based on the respective costs of misdiagnosis and communication. The distributed consistency algorithm allows each agent to form a coalition to reduce its uncertainty or to restore a consistent view of the system state in case some had to act locally with incomplete information at an earlier stage. Moreover, the intertwining of the diagnosis and the repair phases allows an efficient reduction of the diagnosis search-space. The overall reduction of 34% of the Dem@tFactory system down-time associated with a repair extra-cost of 7.2% demonstrate that ADS2 is able to efficiently supervise complex systems under real-life assumptions.

A fully autonomous supervision system is presently not realistic in an industrial context as Humans wants to keep control on what they perceive as critical decisions. ADS2 represents what we see as an acceptable trade-off as the definition of its autonomy degree can be easily accomplished. Thus ADS2 organizes the set of known faults and repairs in several subclasses : the ones whose repair plan can be triggered automatically, and those whose final repair decision rests with a human supervisor. The risk aversion of the users defines the size of these two respective sets. If the confidence in the efficiency of the autonomous supervision of complex and distributed systems is not common today, we believe that the work presented herein provides a step towards this goal.

References

[1] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[2] M. Salehie and L. Tahvildari. Self-adaptive software: Landscape and research challenges. *ACM Trans. Auton. Adapt. Syst.*, 4(2):1–42, 2009.

[3] M.J. Fischer, N.A. Lynch, and M.S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

[4] L. Lamport. Paxos made simple. *ACM SIGACT News*, 32:18–25, 2001.

[5] R. De Prisco, B. Lampson, and N. Lynch. Revisiting the paxos algorithm. *Distributed Algorithms*, pages 111–125, 2000.

[6] Marie-Odile Cordier, Y. Pencolé, L. Travé-Massuyes, and T. Vidal. Self-healability = diagnosability + repairability. In *The 18th International Workshop on Principles of Diagnosis*, volume 7, pages 251–258. Citeseer, 2007.

[7] Philip Koopman. Elements of the self-healing system problem space. In *ICSEWADS03*, 2003.

[8] K. Chmiel, M. Gawinecki, P. Kaczmarek, M. Szymczak, and M. Paprzycki. Efficiency of JADE agent platform. volume 13, pages 159–172. IOS Press, 2005.

[9] Giovanni Betta and Antonio Pietrosanto. Instrument fault detection and isolation: state of the art and new research trends. volume 49, pages 100–107, 1998.

[10] S. Lafortune, D. Teneketzis, M. Sampath, R. Sengupta, and K. Sinnamohideen. Failure diagnosis of dynamic systems: an approach based on discrete event systems. In *Proc. American Control Conference*, volume 3, pages 2058–2071, June 25–27, 2001.

[11] Christos G. Cassandras and Stéphane Lafortune. *Introduction to Discrete Event Systems*. Springer, 2008.

[12] H. Wörn, T. Längle, M. Albert, A. Kazi, A. Brighenti, S. Revuelta Seijo, C. Senior, M A S. Bobi, and JV. Collado. Diamond: distributed multi-agent architecture for monitoring and diagnosis. *Production Planning & Control*, 15:189–200, 2004.

[13] Nico Roos, Annette ten Teije, André Bos, and Cees Witteveen. A protocol for multi-agent diagnosis with spatially distributed knowledge. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, page 7. ACM, 2003.

[14] Peter Fröhlich and Wolfgang Nejdl. Resolving conflicts in distributed diagnosis. In *ECAI Workshop on Modelling Conflicts in AI*, 1996.

[15] W. Nejdl and M. Werner. Distributed intelligent agents for control, diagnosis and repair. *RWTH Aachen, Informatik, Tech. Rep*, 1994.

[16] E. Horvitz and G. Rutledge. Time-dependent utility and action under uncertainty. pages 151–158, 1991.

[17] E.A. Hansen and S. Zilberstein. Monitoring and control of anytime algorithms: A dynamic programming approach. *Artificial Intelligence*, 126:139–157, 2001.