ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-1023

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

Cédric Herpson and **Amal El Fallah Seghrouchni** and **Vincent Corruble**[1]

Abstract. 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. 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. We demonstrate the advantage of our proposal 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

As the complexity of industrial 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 to the supervision system. This requirement has lead to the (re)birth of a research community around the notions of autonomic computing [3] and self-* systems [7]. Our work lies within this context.

Centralized supervision systems are currently the standard in industry. However, they do not perform well in asynchronous contexts [4]. Indeed, communication malfunctions between the supervision system and the different regions of the supervised system delay the repair and do not allow to quickly return to normalcy.

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. To our knowledge, the work of Nejdl *et al* [5] is the only one that addresses the distribution of these two phases. However, they assume communications at no cost. Our proposal does not make such assumptions. The proposed architecture is composed of two mechanisms: A *decision mechanism* and a *coordination and consistency recovery 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 an asynchronous context.

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 [6]. 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 supervised by one agent. The supervision agents exchange information in order to establish a diagnosis and a repair consistent with the alarm they get from the various units of the supervised system.

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 the observations and the messages between agents can be lost but not corrupted. The agents are supposed to be reliable.

To be able to represent any temporal dependencies, each fault is modeled as a t-temporised Petri net. 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). 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. 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.

2.1 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 $Ct(Dimm_{opt})$

[1] University of Pierre and Marie Curie, France, email: firstname.lastname@lip6.fr
[2] The system can only use faults model, *a priori* known or dynamically learned from the system observation.

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 and the delayed one.

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.

2.2 Consensus 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 frequently requires information and knowledge from other supervision agents. It therefore becomes necessary to reach a consensus on the decision to make. In a context of asynchronous and unbounded communication the theorem of Fisher-Lynch-Paterson[2] states the impossibility of guaranteeing the achievement of a consensus between different components. Thus, 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. The multi-Paxos algorithm [1]falls into this category. Using it, each agent is thus able to initiate, integrate or leave a coalition.

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**

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.

Algorithm 1 works in the manner of producer-consumer with the decision-making process introduced in section 2.1. 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 into their own potential inconsistency queue F_{inc}.

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.

3 Experimentals Results and Conclusion

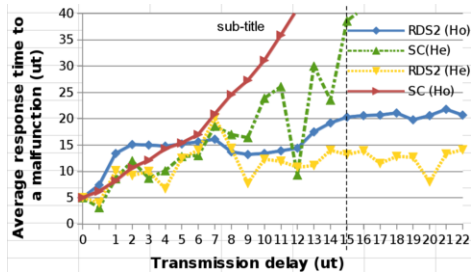

Figure 1. Average response time to a malfunction.

Fig. 1 presents the evolution of the behaviour of our supervision systems $ADS2$ and of a centralised one (SC) regarding the average response time to a malfunction in the case of homogeneous (Ho) and heterogeneous (He) communication links. We used a real data-set provided by our partners to generate the faults and 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.

The vertical bar at t=15ut is the horizon considered by the agents of $ADS2$ for the computation of the delayed decision. Results shows that our architecture is very robust, allowing the supervised system to rapidly recover from failures. 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.
With our industrial fault models data-set, the overall gain regarding the supervised system down-time reach 34%.

In conclusion, we presented the first anytime multi-agent architecture for the supervision of distributed systems 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. The reduction of 34% of our partners system down-time demonstrates that ADS2 efficiently supervises complex systems under real-life assumptions.

REFERENCES

[1] R. De Prisco, B. Lampson, and N. Lynch, 'Revisiting the paxos algorithm', *Distributed Algorithms*, 111–125, (2000).
[2] 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).
[3] J. O. Kephart and D. M. Chess, 'The vision of autonomic computing', *Computer*, **36**(1), 41–50, (2003).
[4] 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, pp. 2058–2071, (June 25–27, 2001).
[5] W. Nejdl and M. Werner, 'Distributed intelligent agents for control, diagnosis and repair.', *RWTH Aachen, Informatik, Tech. Rep*, (1994).
[6] N. Roos, A.T. Teije, A. Bos, and C. Witteveen, 'An analysis of multi-agent diagnosis', in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2*, pp. 986–987. ACM, (2002).
[7] M. Salehie and L. Tahvildari, 'Self-adaptive software: Landscape and research challenges', *ACM Trans. Auton. Adapt. Syst.*, **4**(2), 1–42, (2009).