An Approach for Fault Tolerance in Dynamic Multi-Agent systems

Mounira BOUZAHZAH.
Lire Laboratory, Mentouri University, Constantine, Algeria.
mbouzahzah@yahoo.fr

Ramdane MAAMRI.
Lire Laboratory, Mentouri University, Constantine, Algeria.
rmaamri@yahoo.fr

ABSTRACT

Multi-agent systems offer a decentralized and cooperative vision of the problems solving. That means they are particularly well adapted to dynamic distributed problems mainly new cooperative applications, such as: e-commerce and traffic control. In this paper, we introduce an original hybrid approach for fault resistance in dynamic multi-agent systems. This approach is based on two concepts which are: replication and teamwork. Through this work we propose to evaluate the agent criticality using two different levels: the local level where the agent criticality is calculated according to its local plan of actions and the external level that concerns the different relations between the concerned agent and the other system's agents. The criticality evaluation makes it possible to divide the system's agents into two main groups. The first group called the critical group uses the active replication. The non critical group, however, uses the passive replication. Three other agents are added to the system in order to guarantee the approach efficiency.

KeyWords: agent local criticality, agent external criticality, hybrid approach, action initial criticality, action dynamic criticality.

1. Introduction

Multi-agent systems are prone to the same failures that can occur in any distributed software system. A system faults are classified into to main classes:

- Software faults: those are caused by burgs in the agent program or in the supporting environment.
- Hardware faults: these faults are related to material failures such as: machine crash, communication breakdown…

Several researches are addressed to solve the problem of fault tolerance in multi-agent systems using different strategies. The most important ones are based on the concept of replication. There are different strategies to apply replication, the static strategy which decides and applies replication at design time like in [1], [2] and [3]. The dynamic strategy applies replication during the processing time. This strategy introduces the notion of agent criticality. It is used by [4] and [5]. According to the relation between the agent and its replicas there are two different types of replication. The passive replication that is defined as the existence of one active replica that processes all input messages and transmits periodically its current state to the other replicas in order to maintain coherence and to constitute a recovery point in case of failure [6]. The active replication is defined as the existence of several replicas that process concurrently all input messages [7].

This article introduces an approach for fault resistance in dynamic multi-agent systems. Our approach is based on the criticality calculation using agent's plan to determine the agent local criticality. The interdependence relations are used to calculate the agent external criticality. According to their criticalities agents will be oriented towards two different groups: the critical group managed by an agent called the supervisor, this group uses the active replication strategy. The other group uses the passive replication strategy and it is managed by an agent called the controller.

The whole system is controlled by the decision agent that initializes agents to criticality evaluation and decides which agents are the most critical.

Our approach is general because, first, it is hybrid, it uses the passive and the active replication strategies at the same time; and it uses two levels of criticality evaluation (the local level and the external level). Through this approach we calculate the agent criticality dynamically.

The rest of this paper is organized as follows: section2 covers the related works in the field of fault tolerance. Section3 gives a description to the proposed approach based on dynamic replication. Section4 describes the general architecture of the

system, and finally, Section5 that gives an insight into our future directions and concludes the paper.

2. Review of Related Works

Here we review some important works dealing with fault tolerance in multi-agent systems.

Hagg [2] proposes a strategy for fault tolerance using sentinels. The sentinel agents listen to all broadcast communications, interact with other agents, and use timers to detect agent crashes and communicate link failure. So, sentinels are guardian agents which protect the multi-agent system from failing in undesirable states. They have the authority to monitor the communications in order to react to fault. The main problem within this approach is that sentinels also are subject of faults.

Kumar and al [1] introduce a strategy based on Adaptive Agent Architecture. This strategy uses the teamwork to cover a multi-agent system from broker failures. This approach does not deal completely with agent failures since only some agents (the brokers) or part of them can be replicated.

A strategy based on transparent replication is proposed by [3]. All messages going to and from a replicated group are funneled through the replicate group message proxy. This work uses the passive replication strategy.

These several approaches apply the replication mechanism according to the static strategy which allows replication at design time. But recent applications and mainly those which use the multi-agent systems are very dynamic the fact that makes it too difficult to determine the critical agents at the design time. There are other proposed works that other use the dynamic replication strategy such as:

Guesssoum and al [4] introduce an automatic and dynamic replication mechanism. They determine the criticality of an agent using various data such as: time processing, the role taken by an agent in the system… This mechanism is specified for adaptive multi-agent systems. They focus their work the platform DIMA [8].

Almeida A. and al [9] propose a method to calculate the criticality of an agent in a cooperative system. They use agent plan as the basic concept in order to determine critical agent. This work uses the framework DARX [10].

These two works use the dynamic replication that allows replication at the processing time. This strategy requires the criticality calculation. The agent criticality is defined as the impact of a local failure of an agent on the whole system [11]. The dynamic strategy is more important than the static one when dealing within dynamic applications, but it must use a mechanism able to determine when it is necessary to duplicate agents.

3. The Hybrid Approach

Agents are subject of failure that can cause the whole system failure. We propose an approach to introduce fault tolerance in dynamic multi-agent systems by the use of two main concepts which are: replication and teamwork. Under our approach the two replication strategies are used (active and passive). Since we deal with dynamic multi-agent systems, we will use the dynamic replication, which means that agents are not duplicated at the same time and within the same manner. The question that arises, therefore, is which are the agents to be replicated?

4. The Criticality Evaluation

The agent criticality denoted C_X is defined as the impact of a local failure of the agent X on the dysfunction of the whole system. An agent that causes a total failure of the system will have a strong criticality.

The criticality evaluation in our approach is realized at two main levels:

- The local level: here we determine the agent criticality using its plan of actions.
- The external level: In order to achieve its current goal the agent does not only use its own data but it relies on other agents. So, we try to evaluate the agent external criticality using the relations between agents.

4.1 Agent Local Criticality

In order to calculate the agent local criticality, we defined an agent according to the model proposed by [12]. Each agent is composed of the following elements:

- Goals: the goals an agent wants to achieve.
- Actions: the actions the agent is able to perform.
- Resources: the resources an agent has control on.
- Plans: the plan represents the sequence of actions that the agent has to execute in order to achieve a certain goal.

4.1.1 Agent Plan

We conceder that each agent knows the actions sequence that he has to execute in order to achieve its current goal. Therefore, we propose the use of a graph to represent the sequence of actions called agent's plan. These plans are established for short terms because the environment considered is dynamic. The graph that we use in this work is inspired from that proposed by [9]. The agent plan is represented by a graph where the nodes represent actions and edges represent relations between actions. These relations are the logical functions AND and OR. A node *n* which is connected to *k* other nodes *(n1, n2... nk)* using AND edges represents an action that will be achieved only if all

its following actions are executed. However, a node *n* connected to its *k* followers using OR edges represents an action that is achieved if only one following action is executed. The work proposed in [5] uses a different description concerning the agent plan and it proposes the existence of internal and external actions. However, we are interested to actions which are executed by the agent (local actions), Thus, according to our description an agent X will be represented as follows (Figure 1):

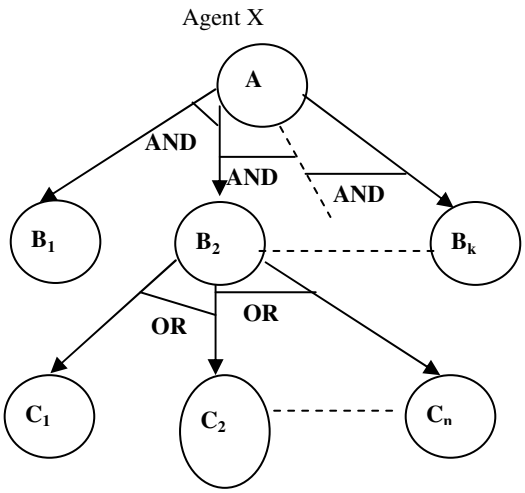

Figure 1. Agent X plan.

4.1.2 Action criticality

In this paper we propose the use of two types of action's criticality: the action initial criticality given by the designer, and the action dynamic criticality calculated according to the agent plan.
Thus, the criticality of an action A denoted C_A is calculated as follows:

C_A = initial criticality + dynamic criticality

$$C_A = C_{IA} + C_{DA}$$

4.1.3 Action Initial Criticality

We admit that a critical agent is the one which executes critical actions. And we propose the following criteria to define the initial criticality of an action:

- An action which can be done by several agents can be regarded as being not too critical, but if an other action is done by few agents it will be regarded as a critical one.
- The number of necessary resources that are required for the execution of an action can be also a factor to determine the initial criticality of an action. When an action requires many resources to be executed, it introduces a strong criticality.
- Hardware data influence, also, the action initial criticality.
- Finally, according to the application field, the designer can determine semantic information that can define the initial criticality of an action.

Thus, at the design time each action A has a value called the initial criticality denoted C_{IA}.

4.1.4 Action Dynamic Criticality

The dynamic criticality of an action denoted C_D is defined as the value attributed to an action according to its position in the agent plan. There is one factor that can influence the action criticality which is the set of its following actions.
We use the function MULTIPLICATION to represent the following actions influence on the considered action when they are connected using AND edges. Since we have indicated that when an action A connected to its followers (B_1, B_2,…, B_k) by AND edges, the achievement of A implies that all its following actions are achieved. If we represent the actions with a group of sets we will have the following result:

$$A = (B_1 \cap B_2 \cap ... \cap B_k).$$
$$C_A = C_{IA} + (C_{B1} * C_{B2} *...* C_{BK})$$

One other function SUM is used to represent the case where one action is connected to its followers by OR edges. If we consider action B_2 (figure 1) connected to its followers (C_1, C_2, …, C_n) by OR edges, in term of sets we will have:

$$B_2 = (C_1 \cup C_2 \cup ... \cup C_n)$$

Thus, B2 criticality is calculated as follows:

$$C_{B2} = C_{IB2} + (C_{C1} + C_{C2} + ... + C_{Cn})$$

An action which has no follower is called a terminal action. The dynamic criticality of a terminal action equals to 0. This means that the criticality of a terminal action equals to its initial criticality.

4.1.5 Agent Local Criticality Calculation

In order to determine the agent local criticality, we admit that each agent knows at an instant *t* the actions sequence which it has to execute to achieve its current goal. The local criticality of agent $C_{L\ agent}$ is calculated as follows:

$$C_{L\ agent} = Sum\ (C_{action1} + ... + C_{action\ n}).$$

This criticality calculation is made directly by the agent.

Example:

Let's calculate the agent local criticality following the agent plan (Figure2):

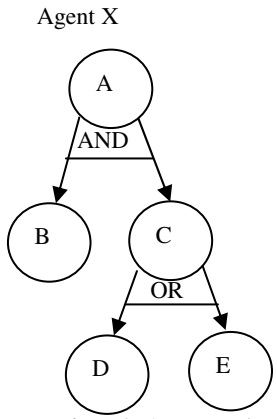

Figure2. Agent X plan

Table1.The initial actions criticalities.

C_{IA}	C_{IB}	C_{IC}	C_{ID}	C_{IE}
2	1	3	5	10

Table1.

$C_A = C_{IA} + (C_B * C_C)$
$C_B = C_{IB}$
$= 1$ B is a terminal action.
$C_C = C_{IC} + (C_D + C_E)$
$C_D = C_{ID} = 5$ D is a terminal action
$C_E = C_{IE} = 10$
$C_C = 18$
$C_A = 20$

The local criticality of agent X:
$C_{LX} = (C_A + C_B + C_C + C_D + C_E)$
$= 54.$

4.2 Agent External Criticality:

According to the agent definition shown in the previous section the agent possesses a set of plans. Each plan is formed of a sequence of actions that the agent has to execute in order to achieve its current goal. These actions do not necessarily belong to the agent set of actions; therefore, an agent may depend on other agents to carry on a certain plans.
There are six different dependence situations identified by [12]. Through this work we are interested to two main dependence relations which are:

- The cooperative relation when an agent infers that he and other agents are depending on each other to realize the same current goal.
- The adoptive relation the situation when an agent infers that he and other agents are depending on each other to realize different current goals.

The relation between agents is defined in our model using the following set:

$$Set = \{T, P, N\}$$

T: represents the relation type, it can be cooperative or adoptive.

P: is the relation weight, here it represents the sum of the initial criticalities of the actions that are executed using this relation:

$$P = Sum\ C_{I\ of\ the\ actions\ executed\ using\ the\ relation}$$

N: the number that represent the agents having the same current goal.
The external criticality in this case is calculated as follows:

$$C_{ex\ agent} = p/N$$

In adoptive case N = 1.

4.3 Agent Criticality

The agent criticality denoted C_{agent} is considered as agent propriety, it is calculated by the agent directly using the following relation:

$$C_{agent} = C_{L\ agent} + C_{ex\ agent}$$

4.4 Determine the Most Critical Agents

Each agent must pass the calculated criticality at the instant *t* to an other agent called the decision agent. This later uses these values to determine the most critical agents. According to usual arithmetic, the median value of N numbers gives an index to divide a unit into two parts. The decision agent uses the following algorithm in order to determine the two groups of agents.

Algorithm: decision
Begin
Sumcriticalities ← 0
For each agent I do
Read $C_{agent\ i}$ /* $C_{agent\ i}$ the criticality of the agent I*/
/* the sum of agents criticalities calculation*/
Sumcriticalities ← Sumcriticalities + $C_{agent\ i}$
For each agent I do
If ($C_{agent\ i}$ >= Sumcriticalities / number of the agents) Then
GT =1
Else
GT=2
/* GT is an agent property, if GT=1 then the agent is affected to the critical group, else it is in the other group*/
End.

Finally, agents are oriented towards two different groups.

4.5 Criticality Re-Evaluation

The criticality calculated in the previous sections is determined at the instant *t*; it must be updated throughout the execution since our system is dynamic. We propose a solution based on two strategies:

- Time strategy: the decision agent has a clock that gives alarms to re-evaluate agents' criticalities at each fixed time interval Δt.

- Event strategy: There are many events that act on the system and caused criticality revision such as: an agent failure, a machine failure.

4.6 Determine the Agents Groups

The concept of teamwork is used by different approaches such as [1] and [2]. Concerning this approach, criticality calculation leads to the creation of two agents' groups. This stage makes it possible to determine a strategy for fault tolerance.

- The critical agents' group: uses the active replication. Each critical agent will have only one active replica called the follower. This later is an agent that has the same plan and executes the same action processed by the critical agent but after the reception of a permission message sent from the supervisor. The supervisor is an agent that guarantees the management of the critical group.
- The no critical agents' group: this group uses the passive replication strategy. Each no critical agent will have only one passive replica. It is the no critical agent that executes all the actions and transmits its current state. If the active agent is lost its replica is activated by an other agent called the controller which is the group's manager.

The criticality revision is done by the decision agent according to two factors: time-driven factor and event-driven factor .When an agent is considered as critical at a given time *t*. It establishes a contract with the supervisor agent. So, the agent will have an active replica. If at the instant $t + \Delta t$, the re-evaluation of the criticality considered the same agent as no critical its contract will be deleted. And one other contract will be established within the controller.

5. System Architecture

In order to guarantee fault tolerance in dynamic multi-agent systems, we have added three agents that allow error detection and data recovering. The general architecture of the system is given by the following diagram (figure3):

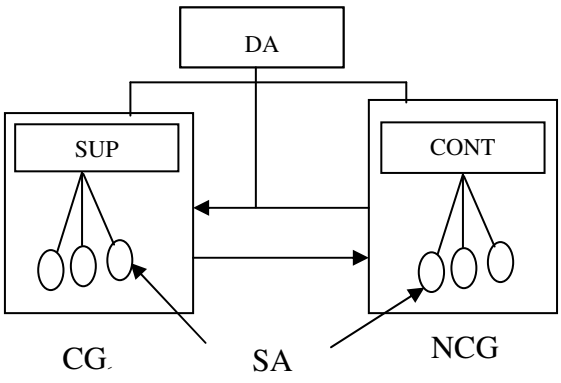

Figure3. The system's architecture.

DA: The Decision Agent.
SUP: The Supervisor.
CONT: The Controller.
SA: The system's Agents.
CG: Critical Group.
NCG: Non Critical Group.

The system consists of the dynamic multi-agent system and the three added agents: the decision agent that controls the whole system, the supervisor which manages the critical group and the manager of the no critical group called the controller.

5.1 The Decision Agent

This agent offers two fundamental services. First it determines critical agents the fact that allows the division of the whole system into two main groups. And it initializes the agents to the process of criticality re-evaluation following the dynamicity of the system.

We use the concept of the sequence diagram [13] in order to represent the decision agent's role as follows (Figure 4).

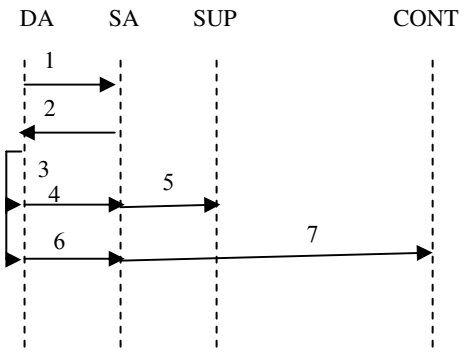

Figure 4. The sequence diagram for the decision agent.

DA: The Decision Agent.
SA: The System's Agent.
SUP: The Supervisor.
CONT: The Controller.
1: The Criticality Evaluation.
2: Pass the Criticality C.
3: Decision.
4: GT= 1.
5: Establish contract with the Supervisor.
6: GT= 2.
7: Establish contract with the Controller.

5.2 The Supervisor

This agent allows the active replication. During execution time, the critical agent transmits periodically its current state to the supervisor, this latter gives permission messages in order to validate the replica's execution.

The supervisor allows also failure detection. This service makes it possible to detect if an agent is still alive and that it does not function in a synchronous environment [14]. The supervisor achieves this service within the use of a clock that initializes the control messages sent to the critical agents. Each activated (critical replica) has a failure – timer which gives the max time used by the agent to answer. If the agent does not give an answer a failure is detected.

Since the failure detection, the supervisor creates a replica and the follower takes up the failed agent. The supervisor's services are represented by the following diagram (Figure 5).

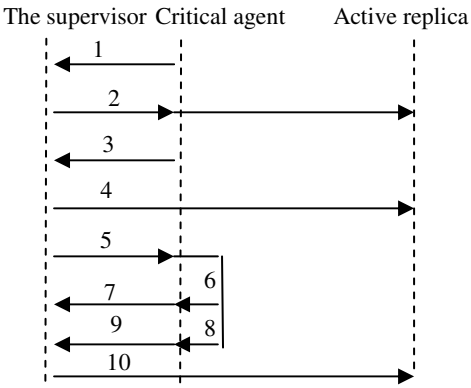

Figure 5. The sequence diagram for the supervisor

1: Establish contract.
2: Active replication process.
3: Current state's message.
4: Permission message.
5: Controlling message.
6: Yes.
7: Answer.
8: No.
9: T > Max Time.
10: Agent recovering.

5.3 The Controller

It is the no critical agent group's manager it allows agent replication using the passive strategy. This agent verifies and detects failure among its group's agents using the same technique employed by the supervisor. Since the detection of failure, the passive replica will be active and an other passive replica will be added. The controller's sequence diagram is represented as follows (Figure 6):

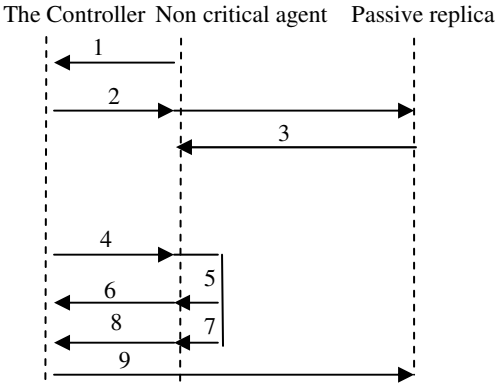

Figure 6. The sequence diagram for the controller.

1: Establish contract.
2: Passive replication process.
3: Current state's message.
4: Controlling message.
5: Yes.
6: Answer.
7: No.
8: T > Max Time.
9: replica activated + Agent recovering.

6. Conclusion

This article proposes a rich approach for fault resistance in dynamic multi-agent systems based on replication and teamwork. We use the two strategies (active and passive) within the existence of one strong replica at one time; this fact allows the decreasing of charges. In order to guarantee failure detection and system controlling three other agents are added.
In further work, we are interesting to propose a more formal model for criticality calculation and to validate our approach trough implementation.

7. References:

[1] S.Kumar, P. R Cohen., H.J. Levesque,"The adaptive agent architecture: achieving fault-tolerance using persistent broker teams", The Fourth International Conference on Multi-Agent Systems (ICMAS 2000), Boston, MA, USA, July 7-12, 2000.

[2] S. Hagg ," A sentinel Approach to Fault Handling in Multi-Agent Systems ", Proceedings of the second Australian Workshop on Distributed AI, Cairns, Australia, August 27, 1996.

[3] A. Fedoruk, R. Deters, "Improving fault – tolerance by replicating agents", Proceedings AAMAS-02, Bologna, Italy, P. 144-148.

[4] Z.Guessoum , J-P.Briot, N.Faci, O. Marin, "Un mécanisme de réplication adaptative pour des SMA tolérants aux pannes ", JFSMA, 2004.

[5] A. Almeida, S. Aknine, et al, "Méthode de réplication basée sur les plans pour la tolérance aux pannes des systèmes multi-agents ", JFSMA, 2005.

[6] M. Wiesmann, F. Pedone, A. Schiper, et al, "Database replication techniques : a three parameter classification". Proceedings of 19th IEEE Symposium on Reliable Distributed Systems (SRDS2000),Nüenberg ,Germany, October 2000 . IEEE Computer Society.

[7] O. Marin,"Tolerance aux Fautes", Laboratoire d'Informatique de Paris6, Université PIERRE & MARIE CURIE.

[8] N. Faci, Z. Guessoum, O. Marin,"DIMAX: A Fault Tolerant Multi - Agent Platform". SELMAS' 06.

[9] A. Almeida, and al, "Plan-Based Replication for Fault Tolerant Multi-Agent Systems", IEEE 2006.

[10] O. Marin, P. Sens,"DARX: A Framework For Tolerant Support Of Agent Software", Proceedings of the 14th International Symposium on Software Reability Engineering, IEEE,2003.

[11] A. Almeida, S. Aknine, et al,"A Predective Method for Providing Fault Tolerance in Multi-Agent Systems",, Proceedings of the IEEE / WIC/ACM International Conference of Intelligent AgentTechnologie (IAT'06).

[12] J. S. Sichman, R. Conte, et al,"A Social Reasoning Mechanism Based On Dependence Networks". ECAI 94, 11th European Conference On Artificial Intelligence, 1994.

[13] M. Jaton,"Modélisation Objet avec UML", cours,chapitre13.
http://www.iict.ch/Tcom/Cours/OOP/Livre/LivreOOPTDM.html.

[14] M. Fischer, N. Lynch, M. Patterson, "Impossibility of distributed consensus with one faulty process". JACM, 1985.