Predictive Fault Tolerance in Multi-Agent Systems: a Plan-Based Replication Approach

Alessandro de Luna Almeida, Samir Aknine, Jean-Pierre Briot, Jacques Malenfant
Université Pierre et Marie Curie-Paris6,UMR 7606
4 Place Jussieu
Paris, F-75005 France
+33 1 44 27 87 94
{Alessandro.Luna-Almeida, Samir.Aknine, Jean-Pierre.Briot, Jacques.Malenfant}@lip6.fr

ABSTRACT
The fact that multi-agent applications are prone to the same faults that any distributed system is susceptible to and the need for a higher quality of service in these systems justify the increasing interest in fault-tolerant multi-agent systems. In this article, we propose an original method for providing dependability in multi-agent systems through replication. Our method is different from other works because our research focuses on building an automatic, adaptive and predictive replication policy where critical agents are replicated to minimize the impact of failures. This policy is determined by taking into account the criticality of the plans of the agents, which contain the collective and individual behaviors of the agents in the application. The set of replication strategies applied at a given moment to an agent is then fine-tuned gradually by the replication system so as to reflect the dynamicity of the multi-agent system. Some preliminary measurements were made to assess the efficiency of our approach and future directions are presented.

Categories and Subject Descriptors
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *Multiagent systems*.

General Terms
Algorithms, Performance, Reliability, Experimentation.

Keywords
Agent, multi-agent system, fault tolerance, adaptation, replication, criticality, plan.

1. INTRODUCTION
In order to prevent that a multi-agent system (MAS) stop working properly due to the occurrence of faults, many fault tolerance approaches have been proposed, notably based on the concept of replication, i.e. creation of copies of a component in distant machines.

In general, it is the responsibility of the designer of the application to identify explicitly what critical components should be made robust and how to parameterize replication. This can be decided either statically before the application starts [4, 7] or in a non-automatic way during the execution of the system [2, 3, 6].

However, those works are not suitable for MAS applications, which can be very dynamic. In fact, it is very difficult, or even impossible, to identify in advance the most critical agents of the application. Moreover, for dynamic applications, a manual control is not realistic, as the application designer cannot monitor the evolution of a distributed application of a significant scale.

In this paper, we will introduce our approach to building reliable multi-agent systems. It is based on the concept of criticality, a value dynamically associated to each agent in order to reflect the effects of its failure on the overall system. This value is automatically calculated using the plans of the agents.

A plan-based fault-tolerant mechanism acts as a promising preventive method since it estimates a more precise value of the criticality and it takes into account the future behaviors of the agents and their influence over the other agents of the society.

We have used the failure detection, naming and localisation services of the DARX framework and we have extended it with an adaptive replication control module which calculates and updates in a distributed way the criticalities and uses the replication service of DARX to provide a suitable replication scheme for every agent.

2. PROBLEM DEFINITION
The fault tolerance problem described in this paper considers a set of agents $S = \{Agent_1, Agent_2, ..., Agent_{na}\}$ that have to complete a set of tasks. For example, consider a set of agents called assistants, which elaborate plans for a patrolling task (see Figure 1). The patrol is generally defined as a task performed by a set of autonomous agents, called patrollers. Each patrolling agent will interact with its corresponding assistant agent in order to discover the sequence of sites which must be visited. A predetermined priority is associated to each site.

While trying to accomplish their tasks, agents can stop executing. In this work, we consider the crash type of failures, that is when a component stops producing output. However, in various cases our

AAMAS'07, May 14-18, 2007, Honolulu, Hawai'i, USA.

solution allows to deal with other types of failures (omission, timing, byzantine).

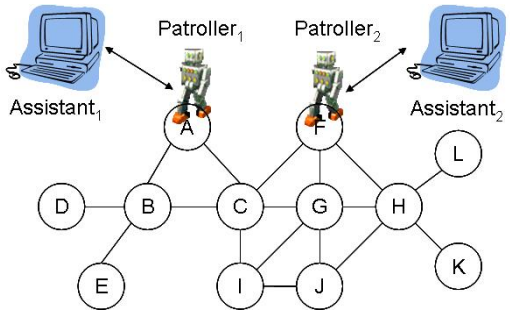

Figure 1. Example of the patrolling task.

To minimize the impact of failures, agents can be replicated. Replicating every agent is not a feasible approach since not only the available resources are often limited, but also the overhead imposed by the replication could degrade performance significantly. The problem consists in finding a replication scheme which minimizes the probability of failure of the most critical agents. This scheme must also be revised over time, considering that the multi-agent execution context of tasks is dynamic and, thus, the criticalities of the agents vary at runtime.

3. OUR PLAN-BASED CRITICALITY ASSESSMENT METHOD

In our approach, we consider that each agent of the system knows which sequence of actions (plan) must be executed in order to accomplish its current tasks. Since unexpected events may occur in dynamic environments, agents usually interleave planning and execution. Consequently, their plans are established just for the short term. We assume that at each given instant of time the agent is executing at most one action.

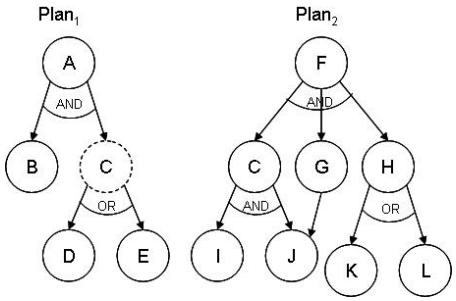

Figure 2. Example of plans of two interacting agents (dashed circles represent external actions).

Inspired by the approach established by [5], we represent the plan of an agent as a directed acyclic AND/OR graph where each node represents an action. The nodes are connected by AND or OR edges. In the example of Figure 2, we show two patrolling plans elaborated by the assistant agents ($Plan_1$ for $Patroller_1$ and $Plan_2$ for $Patroller_2$). For legibility purposes, let us denote the action "*visit the site X*" simply as "*X*". After performing the action *A*, $Patroller_1$ needs to have both *B* and *C* executed in order to accomplish its plan. However, after *C*, only one of *D* or *E* needs to be performed so that $Patroller_1$ accomplishes its plan.

Definition 1: We call an *external action* an action belonging to the plan of an agent which will be executed by others. For example, consider the action *C* belonging to the plan of $Patroller_1$ in Figure 2. Since this action is performed by $Patroller_2$, it is an external action in the current plan of $Patroller_1$.

Definition 2: The *set of children actions* of an action *a* in a plan *p* (denoted by *Children (a, p)*) is the set of actions which are directly connected to the action *a* in the plan *p*. For example, in Figure 2, $Children\ (A, Plan_1) = \{B, C\}$.

Definition 3: A *terminal* action is an action with no child.

3.1 Agent Criticality

The criticality of an agent at any time can be calculated based on the criticalities of the forthcoming actions which belong to its plan. An agent which executes critical actions must be considered critical. In a given time *t*, the criticality of the agent will be given by the relative criticality of the current root of its plans' graph.

Before defining the relative criticality of an action, let's first introduce the concept of absolute criticality and children relative criticality. The *absolute criticality (AC)* of an action is defined without taking into account the current plans of the agents. It is given a priori by the system designer and can be determined in function of a number of factors: number of agents capable of performing the action, resources required for the execution of the action, application dependant information.

In the case of the patrolling task, the absolute criticality of the action *"visit the site s"* is given by the priority of the site *s*.

The *children relative criticality (CRC)* of an action in the context of a plan estimates the aggregation of the criticalities of the children of the action in the plan. Let *a* be one action belonging to a plan *p*, and *f* be the aggregation function (*SUM* for the AND edges or *MEAN* for the OR edges) of the children of *a* in *p*. Then, the children relative criticality of *a* in *p* (*CRC (a, p)*) is formally defined by:

$$CRC(a, p) = f(RC(c_1), RC(c_2), \ldots),\ \forall c_i \in Children(a, p) \qquad (1)$$

Finally, the *relative criticality (RC)* of an action executed by an agent (possibly jointly with other agents) estimates the impact of its failure to the multi-agent system as a whole. The *RC* reflects the importance of the action with respect to the other actions of the system. The *RC* depends on the absolute criticality of the action and on the usefulness of its results to all the agents which depend on it to perform their tasks. For an external action, it is equal to the children relative criticality. For a non-external action *a*, its relative criticality is equal to its absolute criticality plus the sum of the children relative criticalities of *a* in each plan to which it belongs. In other words:

$$RC(a) = AC(a) + \Sigma(CRC(a,p_i)),\ \forall p_i \mid a \in p_i \qquad (2)$$

In dynamic and unreliable environments, actions with a late start time will be executed less probably than actions with an early starting time, since the plans can change or failures can happen. Consequently, we have also refined this strategy for calculating criticalities by considering the expected starting time of actions.

We compute the estimated starting time of the actions using a topological sorting in the graph (top-down) considering the elapsed times of the antecedents and siblings' actions.

To deal with the dynamicity of multi-agent systems, criticalities need to be updated along time. We proposed two main types of strategies to revise the criticality: *time-driven strategies* and *event-driven strategies* (action completion, failure). More details are presented in [1].

3.2 Agent Replication Mechanism

Once estimated the criticality of the agents, one may ask which agents should be replicated and where to deploy the replicas. When addressing the problem of where deploying efficiently the replicas it is essential to take into consideration the failure probability of the replicas. In fact, it is better to have only one replica which will have in the future an almost zero probability of failure than having many replicas which are not reliable.

Hence, we propose the definition of a replica allocation problem which considers the probability of failures and a mechanism which solves this problem in a satisfactory way.

The problem of replica allocation considers a set of agents $S = \{Agent_1, Agent_2, ..., Agent_{na}\}$ and a set of replicas $R = \{r_1, r_2, ..., r_{nr}\}$. N_a and n_r are respectively, the total number of agents and of replicas. We define the *value* of the replica r_k (denoted by v_k), as the probability that it will not crash. A value of one will be attributed to a completely reliable resource, whereas an unreliable one shall have a near zero value.

We define the problem of replica allocation as the optimization problem of finding an allocation of replicas to the agents which maximizes the expected value of the sum of the utility of the MAS. For that, we define an expected global utility function u which evaluates a given allocation. The higher the value obtained by u, the more efficient and fault-tolerant the allocation. Intuitively, an allocation strategy which makes reliable the most critical agents should be well evaluated.

Let g be an allocation of replicas, c_i the criticality of $Agent_i$ and p_i the probability that it will not fail, then:

$$u(g) = c_1 \times p_1 + c_2 \times p_2 + ... + c_{na} \times p_{na} \qquad (3)$$

Thus, the problem of replica allocation consists in finding a replica allocation g_{max} which maximizes the expected global utility function u.

Our agent replication mechanism tries to find a satisfactory replica allocation function. In our mechanism, let V be the sum of the values of all the replicas in the system. Then, an agent $Agent_i$ is allowed to be replicated using a total value of replicas (t_i) proportional to the percentage of its criticality (c_i) with respect to the sum of agents' criticalities (C), as given by the formula:

$$t_i = c_i \times V / C \qquad (4)$$

The system of replication will then allocate to the agent the set of replicas $R_i = \{r_{i1}, r_{i2}, ... , r_{in}\}$, such that $v_{i1} + v_{i2} + ... + v_{in} \leq t_i$ and its probability of failure is minimal among all the possible sets of replicas.

One can apply the same possible strategies used as the agent criticality update policy (time-driven or event-driven) to decide when to re-calculate the values of t_i.

4. CONCLUSION AND PERSPECTIVES

Large-scale multi-agent systems are often distributed and must run without any interruption. To make these systems reliable, we proposed an original predictive method to evaluate dynamically the criticality of agents. Our approach takes profit of the specificities of multi-agent applications and analyses the agents' plans to determine their importance to the system. This approach allows us to obtain a more precise value of the criticality and it takes into account the future behaviors of the agents. The agent criticality is then used to replicate agents in order to maximize their reliability and availability based on available resources.

We are currently conducting experiments, and we believe that our current results are promising. In fact, the algorithms have a negligible overhead and provide a satisfactory reliability.

One of the perspectives of this work is to refine the problem of fault tolerance in multi-agent systems and its evaluation measures, in order to compare the different proposed strategies with an optimal one and using large-scale experiments.

5. REFERENCES

[1] Almeida, A. L., Aknine S., Briot J.P., Malenfant J. A predictive method for providing fault tolerance in multi-agent systems. In *Proceedings of the Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2006)*, Hong Kong, December 2006, 226-232.

[2] Cukier M. et al. AQuA: an adaptive architecture that provides dependable distributed objects. In *Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems (SRDS'98)*, West Lafayette, Indiana, October 20-23, 1998, 245-253.

[3] Favarim, F., Siqueira, F., Fraga, J. S. Adaptive fault-tolerant CORBA components, In *Middleware Workshops 2003*, 144-148.

[4] Fedoruk, A., Deters, R. Improving fault-tolerance by replicating agents. In *Proceedings of the First International Joint Conference on Autonomous Agents And Multi-Agent Systems (AAMAS-02)*, Bologna, Italy, 2002, 737-744.

[5] Horling, B., Lesser, V., Vincent, R., Wagner, T., Raja, A., Zhang, S., Decker, K., Garvey, A. *The TAEMS White Paper*. Dept. of Computer Science, University of Massachusetts at Amherst (UMASS), MA, USA, 1999.

[6] Kalbarczyk, Z., Bagchi, S., Whisnant, K., Iyer, R.K. Chameleon: a software infrastructure for adaptive fault tolerance. *IEEE Transactions on Parallel and Distributed Systems*, 1999, 560-579.

[7] Kraus, S., Subrahmanian, V.S., Cihan, N. Probabilistically survivable MASs. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, Acapulco, Mexico, 2003, 789-795.