Computing the Fault Tolerance of Multi-agent Deployment[1]

Yingqian Zhang [a] Efrat Manisterski [b] Sarit Kraus [bc] V.S. Subrahmanian [c]

David Peleg [d]

[a] *Delft University of Technology, Delft, The Netherlands*
[b] *Bar-Ilan University, Ramat Gan, Israel*
[c] *University of Maryland, College Park, U.S.A*
[d] *Weizmann Institute of Science, Rehovot, Israel*

There have been tremendous advances in the last decade in the theory and implementation of massive multi-agent systems. However, one major obstacle to the wider deployment of multi-agent systems (MASs) is their capability of tolerating failures. MASs that are deployed across a network can quickly "go down" due to external factors such as power failures, network outages, malicious attacks, and other system issues. Protection against such unexpected failures that disable a node is critical if agents are to be used as the backbone for real world applications.

We concern the way *replication* can form the basis of one tool (amongst many that are needed) to prevent a MAS from succumbing to failure. By replicating agents, we hope to improve the *fault tolerance* of a multi-agent system. Fault tolerance and replication techniques have been extensively studied in distributed computing systems, but much less so in the multi-agent systems domain. The faults considered in this paper are those that cause disconnection (or crash) of the nodes in the network where the MAS application resides. The fault model that we consider is one where the failure of each node in the network is represented by a probability. Given such a fault model, agents that locate on the nodes have different probabilities to be unavailable, and therefore the multi-agent system as a whole has some probability of being out of function. The idea of using replication as a fault tolerance method in our work is thus that, when facing failures, at least *one copy* of each agent will continue to reside on a connected, working host computer (node), so that the MAS as a whole can function as a unified application. Furthermore, in this paper, we focus on the problem of measuring the *probability* that a multi-agent system will tolerate the node failure. We call this probability the *survivability* of a MAS system.

Model. We first provide a *probabilistic fault tolerance model* to study the survivability of a given deployment under the assumption of independence of node failures. Consider a multi-agent system $\mathcal{M}$ consisting of a finite set of agents providing one or more services. We assume that $\mathcal{M}$ is deployed over a *fully* connected overlay network $\mathcal{N} = (V, V \times V)$, where V is the set of nodes in the network. Each node $n \in V$ has some fixed amount of resources, denoted $space(n)$, that it makes available to hosting agents in a given multi-agent system. Let $space(a)$ denote the resource requirements of an agent a.

A *deployment* μ is a mapping from V to $2^{\mathcal{M}}$ such that $\mu(n)$ is the set of agents in $\mathcal{M}$ that are deployed at node n in the network. The deployment μ must satisfy the resource constraint. We say that μ is a *valid deployment* if for each $a \in \mathcal{M}$ there is a node $n \in V$ such that $a \in \mu(n)$.

In a network V, any node of V can "go down" or somehow get "disconnected" from the network. We assume each node has some probability of being unavailable (or out of function), due to its disconnection (or crash) from the network. We define the probability of being unavailable of a node by a *disconnect probability function*. In this paper, we assume the node failures are *independent* of one another.

A future failure event F may cause the disconnection of a set of nodes V_F. Such a failure event will give rise to the partial network V', where $V' = V \setminus V_F$. Clearly, given the possibility of node disconnections,

[1] The full version of this paper published as [1].

a partial network V' can possibly materialize in the future for any $V' \subseteq V$ (in case of a failure event F in which the nodes $V_F = V \setminus V'$ get disconnected). Consider a failure event F, and consider the deployment μ restricted to the future network V' for $V' = V \setminus V_F$. The deployment μ may no longer be valid w.r.t. V' (i.e., some agents in $\mathcal{M}$ may not be deployed in any node of V'). We say that the future network V' is valid if this does not happen. We say that the system survives the failure event F (where the set of nodes V_F gets disconnected) if the remaining V' is a valid network. The survivability of μ is obtained by summing up the probabilities of all its possible valid future networks.

Complexity. We show that even assuming independence, the problem of computing the survivability of a given deployment is at least NP-hard. Moreover, it is also hard to approximate up to a factor of $2^{|V|^{1-\epsilon}}$, where $|V|$ is the number of the nodes in the network. In addition, we show that the complexity of finding the most survivable deployment is NP-hard. We also show that for any polynomial approximation to find a sub-optimal deployment, there will be instances in which the survivability of the most survival deployment is 1 but the algorithm returns a deployment with a survival probability of 0. Thus, any polynomial-time approximation algorithm is guaranteed to find at least one arbitrarily bad solution.

Algorithms. We present two algorithms to accurately compute the probability of survival of a given deployment. One of these algorithms is exponential in the number of agents, while the other is exponential in the number of nodes in the network. Thus, if one of these quantities is small, we can use this algorithm to accurately compute the survival probability of a deployment.

As the exact algorithms are too expensive for real-world applications, we propose heuristics to compute the survivability of a deployment. We are interested in finding lower bounds, which will allow us to guarantee that a given deployment has a survival probability that exceeds some threshold. We develop five such heuristic algorithms to compute the survivability of a given deployment, i.e., the tree-based algorithm, the disjoint based algorithm, the group based algorithm, the anytime algorithm, and the split algorithm.

Experiments. Finally we compare the performance of our algorithms in terms of computation time and solution quality, i.e., how close the solution found by the heuristic is to either the correct solution or to a bound on the solution in cases where the solution cannot be accurately computed. We did this under five different environmental settings. Our results show that all heuristics give a good approximation (with a ratio over 0.85). Especially, the split algorithm has demonstrated a relatively stable performance in terms of quality in all problem instances, and consistently returned the solutions that are over 95% of the optimal performance. Nonetheless its running time is dependent on the setting and the problem size. In contrast, we show that the performance of other heuristics, namely the tree-based algorithm, the disjoint based algorithm, the group based algorithm, and the anytime algorithm, vary greatly with the environment settings. Therefore, each of these algorithms is appropriate for certain settings, but not for others. Based on the experimental results, we identify the conditions under which one algorithm is preferable to another so that MAS applications have some foundation upon which to base a decision about which algorithm to use.

Future work. There are many other interesting directions for future work that are related to this topic. In many real-world applications, it is essential to maintain a *minimal level of survival*. Thus, one variant of the current approach would be to develop algorithms for deployments that meet such minimal survival requirements. Another issue is the trade-off between the survivability of a multi-agent system and its performance. Ensuring survivability could be costly. Consequently this comes at the cost of actually providing the services the multi-agent system is supposed to provide. Yet another important issue is "gaming" the system. For example, suppose the methods described in this paper are used to implement MAS security for an application. A user who knows that the techniques of this paper are used in the system can try to utilize this knowledge in order to break security. This leads to a game theoretic framework whereby we need to reason about how an adversary would make use of such knowledge in order to break the system.

References

[1] Yingqian Zhang, Efrat Manisterski, Sarit Kraus, V. S. Subrahmanian, and David Peleg. Computing the fault tolerance of multi-agent deployment. *Artificial Intelligence*, 173(3-4):437–465, 2009.