A Simulation Framework for Evaluating Distributed Reputation Management Systems

V. Agate, A. De Paola, G. Lo Re, M. Morana

In Proceedings of the 13th International Conference on Distributed Computing and Artificial Intelligence

http://link.springer.com/chapter/10.1007/978-3-319-40162-1_27

A Simulation Framework for Evaluating Distributed Reputation Management Systems

Vincenzo Agate, Alessandra De Paola, Giuseppe Lo Re, and Marco Morana

Università degli Studi di Palermo, 90128 Palermo, Italy,
{alessandra.depaola, giuseppe.lore, marco.morana}@unipa.it,
home page: http://www.dicgim.unipa.it/networks/ndslab/

Abstract. In distributed environments, where interactions involve unknown entities, intelligent techniques for estimating agents' reputation are required. Reputation Management Systems (RMSs) aim to detect malicious behaviors that may affect the integrity of the virtual community. However, these systems are highly dependent of the application domain they address; hence the evaluation of different RMSs in terms of correctness and resistance to security attacks is frequently a tricky task. In this work we present a simulation framework to support researchers in the assessment of a RMS. The simulator is organized in two logic layers where network nodes are mapped to system processes that implement the interactions between the agents. Message Passing Interface (MPI) is used to enable communication among different distributed processes and provide the synchronization within the framework. A case study addressing the simulation of two different attacks to a RMS is also presented.

Keywords: simulation framework, distributed reputation management, multiagent system

1 Introduction

Nowadays many distributed applications are based on unknown agents that interact to each other in a cooperative way. However, malicious or selfish agents may act in a unpredictable way leading to a detriment of the performance of the distributed system. Reputation Management Systems (RMSs) are designed to detect and discourage antisocial behaviors that negatively affect the whole community. The role of a RMS is crucial in totally distributed systems, where a centralized entity capable of coordinating the interactions among agents is missing. In such a scenario, distributed RMSs allow each member of the community to contribute in estimating the reputation of the agents so as to reward those that act honestly and cooperatively.

Since reputation management systems are frequently designed to fit a specific application scenario, researchers are used to design ad-hoc simulators to evaluate the performances of a single RMS. In this work we move a step forward by presenting a novel simulation framework that aims to support researchers in the evaluation of the effectiveness and resistance to security attacks of a generic

Table 1: Comparison between simulation frameworks.

	Proposed Approach	**ART**	**TREET**	**[1]**
Independence from application scenario	X			X
Simulation of security attacks	X		X	X
Specification of event pattern	X			
Simulation of agent interactions	X	X	X	

RMS. The simulator we propose here is independent of a specific application scenario, so it can be adopted in heterogenous contexts ranging from peer-to-peer applications for file sharing [16], e-Commerce frameworks [4] to service oriented architectures [14, 2].

In order to separate the high-level representation of the network from the underlying management of the simulation, the framework is organized in two logic layers, i.e., the *reputation layer* and the *simulation layer*. The overall RMS is modeled as a synchronous distributed algorithm, according to the principles described in [15]. Such assumption does not affect generality and correctness in the evaluation process and allows to straightforwardly verify the RMS's ability to correctly evaluate agents reputation and its resistance to security attacks.

The remainder of the paper is organized as follows: related work is reported in Section 2. The architecture of the simulation framework is described in Section 3, whilst Section 4 presents a case study on the use of the simulator to model the behavior of a RMS and two attacks to its security. Conclusions are discussed in Section 5.

2 Related Work

In distributed systems, where a central authority is missing, and the agents cooperatively estimate the reputation of the other members of the community, RMSs are susceptible to different type of security attacks [10]. Malicious agents may aim at different goals: *self-promoting*, to exploit system vulnerabilities in order to increment their own reputation; *slandering*, to decrease the reputation of a "victim"; *whitewashing*, to "clean" a bad reputation avoiding the negative effects of the disincentive system; and *denial of service*, to block the functioning of the system. To be more effective, attacks may follow an orchestrated plan that requires several malicious agents to work together.

Some testbeds and simulators have been proposed for assessing the performances of a RMS, but none of them allows for large-scale simulations of the behavior of a RMS under security attacks. ART (Agent Reputation and Trust) [5] is a popular simulation testbed in the field of multi-agent systems and allows to apply several evaluation metrics, and to define competitions in which different strategies can be combined and compared. TREET [12] allows to measure the resistance of a RMSs to some attacks (e.g., reputation lag, proliferation, and value imbalance), but does not consider common attacks to distributed systems,

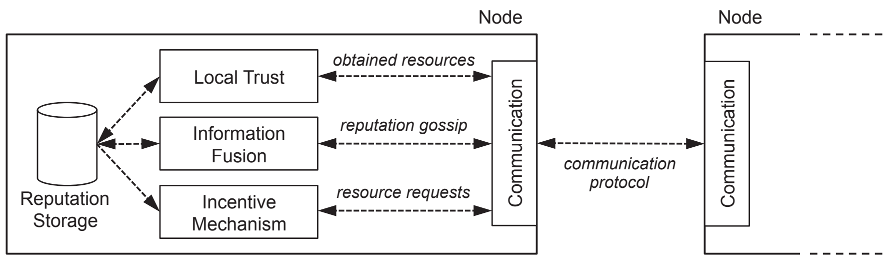

Fig. 1: The components of a distributed RMS. Each node privately performs local trust evaluation and information fusion algorithms; gossip protocol and incentive mechanism determine the interactions with other agents.

such as whitewashing and self-promoting. TREET overcomes many limitations of ART, allowing agents to randomly join or leave a running simulation. The testbed presented in [1] models existing RMSs as a set of transformations on a graph that represents transactions and trust among agents. Even if this system allows to evaluate security resistance to slandering and self-promotion attacks, it does not simulate agent interactions that are crucial to perform large-scale simulations where agents may modify their behavior. The main differences between the simulation framework proposed here and the discussed approaches are outlined in Table 1.

3 Simulation Framework Architecture

In order to design a generic framework, we identified the main components that are common to most of the distributed RMSs proposed in the literature (Fig. 1): (i) a *local trust* evaluation mechanism, used for assessing the behavior of the nodes involved in direct interactions, (ii) a *gossip protocol*, which propagates the local trust to other nodes of the network, (iii) an *information fusion* mechanism to merge information gathered through the gossip protocol with the local trust, and obtain the reputation values, and (iv) a *disincentive mechanism* which exploits reputation values in order to discourage antisocial behaviors.

In our framework we adopt a synchronous model to represent a RMS, which implies that the nodes of the reputation network act simultaneously. Such simplification allows to disregard many details (e.g., unpredictable communication delay or unfair work overload) that, although irrelevant to the assessment of the RMS, may affect the simulation. The simulation proceeds in rounds, each consisting of seven steps: (1) generate resource requests according to the current state, i.e. current view of other nodes reputation, (2) send resource request messages to the destination nodes, (3) elaborate incoming requests and determine whether provide a positive resource response according to the incentive mechanism and the current state, (4) send resource response messages to the destination neighbors, (5) update the current state according to the local trust

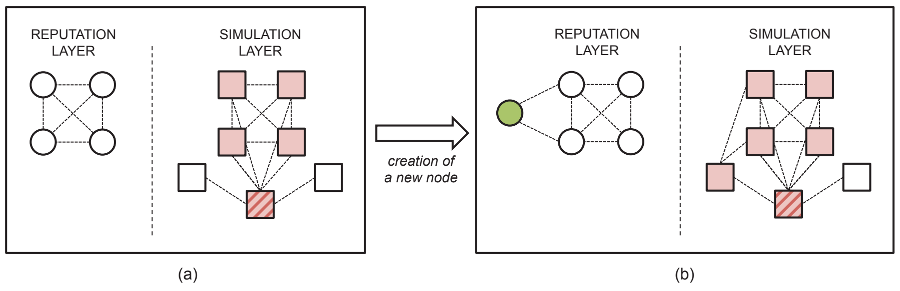

Fig. 2: Creation of a new node as seen at the reputation and simulation layer. (a) The agents (circles) are modeled by active processes (squares) managed by a controller (striped square). (b) When a new node (green circle) is connected to the network, one of the inactive processes (white squares) is woken up and communicates with other active processes.

evaluation, (6) send updated reputation values to the neighbors according to the gossip protocol, and (7) update the information fusion mechanism.

From a logical point of view, the simulation framework is organized in two different layers so as to to separate low-level functionalities, necessary for driving the simulation, from high-level routines, concerned with the tasks of the RMS. A similar architecture was adopted in a previous work where we addressed the design of a simulator for Wireless Sensor Networks in a distributed scenario [13].

The topmost layer, namely the *reputation layer*, is made of nodes connected to each other according to a specific network topology. At this level of abstraction, the reputation network is shown as totally distributed, non-centralized, and a set of high-level configuration utilities are offered to the user. At the *simulation layer*, each agent is mapped to a different process. When the simulator is started, a set of *inactive* processes (see Fig. 2-a) is created. Then, if a new node is added to the network, one of the *inactive* processes is awoken by the simulation manager, i.e. a process called *controller*, and linked to the other processes involved in the simulation (Fig. 2-b).

Since each process can run on a distinct remote host, we chose to adopt the Message Passing Interface (MPI) to enable communication among different distributed processes. MPI provides a protocol for parallel message-passing in distributed scenarios where processes exchange data through cooperative operations. The synchronization is managed by means of blocking and non-blocking, point-to-point or collective, communication primitives that guarantee safe access to shared data. All processes use the thread-safe **MPI_Iprobe()** routine to manage incoming messages. Messages are sent and received by means of the nonblocking functions **MPI_Isend()** and **MPI_Irecv()** respectively, whilst **MPI_Barrier()** is used to synchronize all processes within the coordinator.

4 Case Study

In order to prove the effectiveness of our framework in evaluating a general reputation management system, we considered as case study a simple RMS inspired by [11] and [3].

The local trust component we adopted is a based on that used in EigenTrust [11], one of the best known RMSs for P2P networks. Each agent i stores the number of satisfactory, $sat(i,j)$, and unsatisfactory, $unsat(i,j)$, transactions between agents i and j. Then, the local trust s_{ij} is defined as the difference between such values, i.e., $s_{ij} = sat(i,j) - unsat(i,j)$. The simulation framework supports the user by providing the number of requests sent by each agent, together with the number of negative and positive feedback obtained by other agents. These values are updated at each time step; thus, the researchers can choose how much information they want to use, e.g., all values obtained since the beginning of the simulation or the average value computed in a sliding window.

The gossip protocol can be defined by means of a set of routines that allow for obtaining information about the reputation network and for supporting communication among agents. In particular, it is possible to get the list of the current neighbors, send unicast messages to specific neighbors, and send broadcast messages to the whole neighborhood. The gossip protocol deployed in the case study assumes that agents send their reputation values to all their neighbors. At the end of this phase, each agent knows the opinion of all its one-hop neighbors about the reputation of its two-hop neighborhood. Reputation information gathered so fare is then merged during the information fusion phase, inspired by the work proposed in [3]. Here each agent merges only information coming from reliable agents, i.e., those whose reputation is beyond a given threshold τ. Merged information is weighted with the reputation of the "gossiper" agents, and the resulting reputation value r_{ij} is a linear combination of this weighted mean and the normalized local trust c_{ij}, as specified by the following equation:

$$r_{ij}(t) = (1-\beta) * c_{ij}(t) + \beta * \frac{\sum\limits_{k \in K} r_{ik}(t-1) * r_{kj}(t-1)}{\sum\limits_{k \in K} r_{ik}(t-1)}, \tag{1}$$

where β is a coefficient in $[0,1]$ and K is the set of reliable agents:

$$K = \{k : r_{ik}(t-1) \geq \tau\}. \tag{2}$$

Different attacks can be performed by defining the behavior of the agents through a set of configuration files, and by specifying the number of nodes, the cooperativeness degree of each agent, and the topology of the reputation network. It is important to guarantee that the network is not partitioned in islands, and that each node has an adequate number of neighbors. The current version of the simulator supports two of the most relevant attacks to distributed RMSs, i.e., slandering and whitewashing attacks.

Slandering attacks aim to change the reputation of other agents by disseminating false negative feedbacks in order to obtain some advantage. The simulation framework can be used to assess slandering attack by selecting a victim

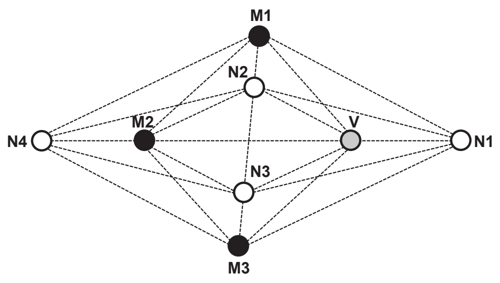

Fig. 3: The network topology used for simulating a slandering attack performed by the malicious nodes M_i against the victim node V.

node V, and a set of M malicious nodes, adjacent to V. These nodes are programmed to share with their neighbors a fake reputation value of the node V, as shown in Fig. 3.

During a whitewashing attack, a malicious node exploits system resources until its reputation goes under an acceptable threshold; then, it leaves and rejoins the community with a new identity associated with a default reputation value. The scenario considered for evaluating whitewashing attacks is shown in Fig. 4. Here, we suppose that a node A, with a low reputation value, wants to duplicate itself to keep exploiting the system resources. In such a situation, the corresponding *active process* P_A sends a duplication request to the *controller* P_C. Then, P_C awakes one of the *inactive* processes P_E and change its state creating the process $P_{A'}$. This cloned process is initialized with the same adjacency list and behavior of P_A, e.g., if the node A was connected to B, then the process $P_{A'}$ will be connected to the process P_B.

At the end of a simulation, the framework provide the detailed trends of reputation and obtained resources over time for all agents. The comparison of the estimated reputation for a given agent with its cooperativeness degree allows to evaluate the average accuracy rate of a RMS, while the average utility provides information about the quality of service experienced by the end user. The slandering attack may be assessed by analyzing the reputation of the victim agent as observed from the point of view of a neutral agent. If the RMS is vulnerable to such class of attacks, the reputation of the victim node should decrease over time, in spite of its honest behavior, because of the effect of the false bad opinions gossiped by malicious agents. The whitewashing attack may be assessed by comparing the aggregate resources obtained through all the false identities of the malicious agent, with respect to those obtained only by its original identity; such value should present an oscillatory trend over time, due to the connection of new identities.

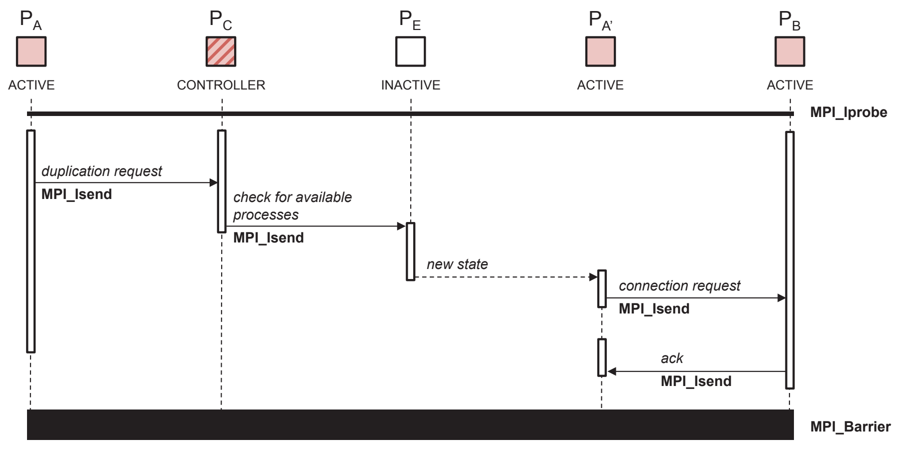

Fig. 4: High-layer interactions and MPI calls used to duplicate a node A, associated with a process P_A.

5 Conclusions and Future Work

In this work we presented a two-layer simulation framework that can be used by researchers for an early evaluation of the performances of a Reputation Management System. The two logic layers allow to separates the reputation management techniques from the software routines that actually drive the simulation. Each agent of the *reputation layer* is mapped to a process running at the *simulation layer*, where MPI interfaces are used to enable communication among different distributed processes. The simulator offers several utilities that can be used to define both the topology of the simulation network, and all the parameters needed to implement the behavior of the simulated agents.

We presented a case study where an ad-hoc RMS was modeled and tested against two different attacks. The results we obtained demonstrate the suitability of the proposed framework for providing detailed information required to perform deep analyses on the performance of the considered RMS.

Future experiments will be performed in order to provide an in-depth evaluation of the framework scalability for large-scale simulations, in terms of memory occupancy, computational and communication complexity, also exploiting advanced techniques for an efficient allocation of processes [8, 9]. Moreover, we are considering the adoption of the simulator to model the interactions within large-scale social networks, e.g., Twitter [7, 6].

Acknowledgement

Work partially supported by the Italian Ministry of Education, University and Research on the "StartUP" call funding the "BIGGER DATA" project, ref. PAC02L1_0086 CUP: B78F13000700008.

References

1. Chandrasekaran, P., Esfandiari, B.: Toward a testbed for evaluating computational trust models: experiments and analysis. Journal of Trust Management 2(1), 1–27 (2015)
2. Crapanzano, C., Milazzo, F., De Paola, A., Lo Re, G.: Reputation management for distributed service-oriented architectures. In: Proc. of the 2010 Fourth IEEE Int. Conf. on Self-Adaptive and Self-Organizing Systems Workshop (SASOW). pp. 160–165 (2010)
3. De Paola, A., Tamburo, A.: Reputation Management in Distributed Systems. In: Proc. of the 3rd Int. Symp. on Communications, Control and Signal Processing (ISCCSP). pp. 666–670 (2008)
4. Fouliras, P.: A novel reputation-based model for e-commerce. Operational research 13(1), 113–138 (2013)
5. Fullam, K.K., Klos, T.B., Muller, G., Sabater, J., Schlosser, A., Topol, Z., Barber, K.S., Rosenschein, J.S., Vercouter, L., Voss, M.: A specification of the agent reputation and trust (ART) testbed: experimentation and competition for trust in agent societies. In: Proc. of the fourth int. joint conf. on Autonomous agents and multiagent systems. pp. 512–518 (2005)
6. Gaglio, S., Lo Re, G., Morana, M.: Real-time detection of Twitter social events from the user's perspective. In: Proc. of the 2015 IEEE Int. Conf. on Communications (ICC). pp. 1207–1212 (2015)
7. Gaglio, S., Lo Re, G., Morana, M.: A framework for real-time Twitter data analysis. Computer Communications 73, Part B, 236 – 242 (2016)
8. Genco, A., Lo Re, G.: A recognize-and-accuse policy to speed up distributed processes. In: Proceedings of the thirteenth annual ACM symp. on Principles of distributed computing. p. 386. ACM (1994)
9. Genco, A., Lo Re, G.: The egoistic approach to parallel process migration into heterogeneous workstation network. Journal of Systems Architecture 42(4), 267–278 (1996)
10. Hoffman, K., Zage, D., Nita-Rotaru, C.: A survey of attack and defense techniques for reputation systems. ACM Computing Surveys (CSUR) 42(1), 1 (2009)
11. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The Eigentrust algorithm for reputation management in P2P networks. In: Proc. of the 12th Int. Conf. on World Wide Web. pp. 640–651 (2003)
12. Kerr, R., Cohen, R.: Smart cheaters do prosper: defeating trust and reputation systems. In: Proc. of The 8th Int. Conf. on Autonomous Agents and Multiagent Systems-Volume 2. pp. 993–1000 (2009)
13. Lalomia, A., Lo Re, G., Ortolani, M.: A hybrid framework for soft real-time wsn simulation. In: Proc. of the 13th IEEE/ACM Int. Symp. on Distributed Simulation and Real Time Applications, 2009 (DS-RT '09). pp. 201–207 (2009)
14. Lee, J., Zhang, J., Huang, Z., Lin, K.J.: Context-based reputation management for service composition and reconfiguration. In: Proc. of the 2012 IEEE 14th Int. Conf. on Commerce and Enterprise Computing (CEC). pp. 57–61 (2012)
15. Lynch, N.A.: Distributed algorithms. Morgan Kaufmann (1996)
16. Marti, S., Garcia-Molina, H.: Taxonomy of trust: Categorizing P2P reputation systems. Computer Networks 50(4), 472–484 (2006)