Ph.D. Thesis

Vincenzo Agate

Reputation Management Algorithms in Distributed Applications

For my family and for my love, for everything you do

Abstract

Nowadays, several distributed systems and applications rely on interactions between unknown agents that cooperate in order to exchange resources and services.

The distributed nature of these systems, and the consequent lack of a single centralized point of control, let agents to adopt selfish and malicious behaviors in order to maximize their own utility. To address such issue, many applications rely on Reputation Management Systems (RMSs) to estimate the future behavior of unknown agents before establishing actual interactions.

The relevance of these systems is even greater if the malicious or selfish behavior exhibited by a few agents may reduce the utility perceived by cooperative agents, leading to a damage to the whole community. RMSs allow to estimate the expected outcome of a given interaction, thus providing relevant information that can be exploited to take decisions about the convenience of interacting with a certain agent. Agents and their behavior are constantly evolving and becoming even more complex, so it is increasingly difficult to successfully develop the RMS, able to resist the threats presented.

A possible solution to this problem is the use of agent-based simulation software designed to support researchers in evaluating distributed reputation management systems since the design phase.

This dissertation presents the design and the development of a distributed simulation platform based on HPC technologies called DRESS. This solution allows researchers to assess the performance of a generic reputation management system and provides a comprehensive assessment of its ability to withstand security attacks. In the scientific literature, a tool that allows the comparison of distinct RMS and different design choices through a set of defined metrics, also supporting large-scale simulations, is still missing.

The effectiveness of the proposed approach is demonstrated by the application scenario of user energy sharing systems within smart-grids and by considering user preferences differently from other work.

The platform has proved to be useful for the development of an energy sharing system among users, which with the aim of maximizing the amount of energy transferred has exploited the reputation of users once learned their preferences.

Acknowledgments

I want to thank all the people who helped me along the way, both professionally and personally. First, I would like to sincerely thank my advisor, Prof. Salvatore Gaglio, for always sharing his knowledge and experience with me.

I wish to express my deepest gratitude to Prof. Giuseppe Lo Re, for everything he has done for me, for being always there when I needed help with my work and with my personal problems. I learned more from him than from anyone else since the beginning of my Ph.D. research. I will forever be in debt.

I owe a huge debt of gratitude to Dr. Marco Morana and Dr. Alessandra De Paola for constantly encouraging and inspiring me, and for all their comments and suggestions to improve my research projects. Without their dedication and hard work, none of this would have been possible.

Thanks to Prof. Simone Silvestri, your brilliant insights have been a tremendous help, and I sincerely appreciate the time you took to guide me and my work.

I will be forever grateful to my dear friend Pierluca Ferraro for his invaluable help over the past years and for being so incredibly supportive. When I didn't feel good enough, you always trusted me. You are the best.

Special thanks to all my amazing colleagues and friends at the NDS Research Group and CPS Lab. It is always a great pleasure and honor to work with all of you.

I am truly lucky to be surrounded by friends who always support me. I cannot properly express what you all mean to me.

Last but not least, I can't begin to express how thankful I am to my family, and especially to my parents and my sister, who are always amazingly helpful and patient in dealing with me. Thank you for your unwavering and unconditional love and support. To my love, thank you from the bottom of my heart for your unfailing enthusiasm, laughs and encouragement. I don’t know what I would do without you. You make all the hard work worth it.

Contents

List of Figures

List of Tables

Glossary

AMI	Advanced Metering Infrastructure
ART	Agent Reputation and Trust
DC	Direct Current
DR	Demand Response
DRESS	Distributed RMS Evaluation Software
ESS	Energy Sharing System
EWMA	Exponentially Weighted Moving-Average
GAP	Generalized Assignment Problem
HPC	High Performance Computing
MAS	Multi-Agent System
MILP	Mixed Integer Linear Programming
MPI	Message Passing Interface
PV	PhotoVoltaics
QoS	Quality of Service
RMS	Reputation Management System
SCN	Single Core Node
UPL	User Preference Learning
VPP	Virtual Power Plant

Chapter 1

Introduction

The pervasive use of the Internet in many aspects of our lives, such as in commerce, social interactions and the exchange of information and services has led to the establishment of a new generation of increasingly complex and advanced applications. Most of the software tools through which users access the services available on the network are based on agents who, while maintaining autonomy and individuality, cooperate by solving shared problems. The early architectural paradigm that exploited centralized computing platforms has been replaced by distributed and strongly interconnected systems. This scenario, while offering fascinating opportunities, presents numerous problems, among which one of the most relevant is the need to estimate the reputation of the agents involved in the interactions, in order to ensure a safe and efficient collaboration between the participants in the network. In fact, in a distributed system agents often have to interact without having a prior mutual knowledge and without having sufficient information to establish secure exchanges.

One of the most accepted solutions to this problem is the use of Reputation Management Systems (RMSs). Typically, reputation-based systems of trust calculate values of reliability of agents, whether they are software or humans, on the basis of evaluations issued by participants at the end of direct interactions, also taking into account past history. The applications that use these systems aim to support the participants in identifying dangerous agents, who adopt malicious and tampering behavior, in order to protect the entire system from possible abuse.

Research into trust and reputation management systems is highly interdisciplinary and cuts across a range of sectors such as networking and communication, data management and information systems, e-commerce and service computing, artificial intelligence and social sciences.

In the recent past, many of the studies carried out in the field of reputation-based trust systems have made enormous progress, both in defining solid theoretical foundations for the implementation of new methods and in verifying the application of such systems to real scenarios. To better understand the importance of such systems in everyday life, it is enough to remember that, in the e-commerce sector, leading companies such as Amazon, eBay and NetFlix use reputation-based mechanisms to ensure the quality and reliability of the services offered (Ayday and Fekri, 2012). Many research efforts have been made in the field of RMS, especially in the definition of specific metrics to obtain reliable evaluations and in the creation of distributed algorithms for the propagation of reputation information. Although a rich literature on RMS is available, the major limitation of many of the solutions presented is the strong dependence of each of them on the specific application scenario considered, and therefore the consequent lack of a general formulation that can adapt to different scenarios. Very little has been done to integrate trust models related to different communities and contexts, in order to improve the robustness and reliability of data-intensive and large-scale distributed applications. Nowadays it is necessary to face a precise study on the problem of reputation management in distributed applications and this thesis wants to offer a contribution with the aim of filling the lack of generality of existing solutions. One of the aspects that cannot be ignored is the robustness of RMS against cybersecurity attacks. Most of the existing distributed algorithms for the implementation of attack-resistant RMS often are inadequate to meet the requirements of generality and abstraction from the specific application context. The need for new solutions becomes evident and is still an open problem.

The aim of this thesis is to encourage the definition of innovative techniques and models that can contribute to the creation of advanced reputation management systems, distributed and adaptive. The research activity has been focused on the creation of a simulation framework that allows an RMS designer to define new methods that can be integrated into existing RMS, or combined in order to create innovative RMS. To achieve this goal, an innovative simulation platform was designed and built to evaluate the effectiveness and robustness of different RMS solutions. The advanced model of simulation that is proposed, uses the technologies of High Performance Computing (HPC) to allow the inspection on a large scale of the characteristics of an RMS, thanks to the high degree of parallelism achievable. The development of simulation tools that allow to evaluate the performance of RMS regardless of a specific application scenario, its robustness against security attacks, and that also allow large scale simulations, has no precedent in the scientific literature.

In order to show DRESS functionalities and to demonstrate the high degree of configurability that can be achieved, part of the efforts have focused on identifying all the actions that a researcher must take step by step to implement their application scenario and their RMS. For this reason it has been realized a case study with all the characteristics of a distributed reputation system, to show how to configure a particular RMS to be analyzed, and then how to run the simulation using the features provided by the simulation tool. In this thesis an analysis of the trend of the reputation agents will be described first, then an analysis of the accuracy and a comparison between the same case study will be presented, modifying some policies within the system under examination. A security analysis and an assessment of scalability of the platform will be presented.

The experimental results show the usefulness of the simulation platform in the application scenario of electricity distribution within smartgrid contexts, in particular in the integration of technologies for reputation estimation in the context of Smart Grid Security. Renewable, heterogeneous and distributed energy resources are the future of energy systems, as foreseen by the recent paradigm of Virtual Power Plants (VPP). The generation of residential electricity, for example through photovoltaic panels, plays a fundamental role in this paradigm, where users will also have the opportunity to exchange energy resources by participating in a system of energy sharing. Also in this context the reliability of the user has a leading role, and succeeding in estimating the most participative users has turned out to be a challenging task. The final part of this dissertation focuses on the application scenario of such sharing systems and, unlike previous approaches, considers realistic user behavior, taking into account preferences and the level of involvement in energy transactions. With the aim of maximizing the amount of energy exchanged among smartgrid users, trying to minimize unsuccessful energy exchanges once the nature of the users was known, the problem of matching energy resources is addressed, as a MILP (Mixed Integer Linear Programming) problem, and it is shown that the problem is NP-Hard. Since the solution to this problem requires knowledge of the user's behavioral model, in which reputation estimation is an indicator of the degree of preference, involvement and participation, a heuristic approach based on reinforcement learning and on a trade-off between exploration and exploitation for the learning of this model is proposed, while optimizing the system's performance. Comparison with state of the art approaches using realistic simulations based on real tracks shows that the proposed method exceeds existing schemes in terms of different efficiency metrics.

1.1 Motivations and Goals

Different systems and distributed applications are based on interactions between unknown agents that cooperate to exchange resources and services. The distributed nature of these systems, and the possible lack of a single centralized control point, allows agents to adopt selfish and harmful behaviors to maximize their usefulness. To tackle this problem, several distributed applications employ Reputation Management Systems (RMS) to estimate the future behavior of unknown agents before establishing actual interactions. The relevancy of these systems is even higher if the malicious or selfish behavior shown by a few agents can reduce the perceived usefulness of the cooperative agents, causing damage to the entire community. RMS enables the prediction of the expected outcome of a given interaction, thus providing relevant information that can be used to make decisions about the convenience of interacting with a certain agent.

The presence of a RMS represents an incentive to cooperative and honest behaviors, since a perceived high reputation generally corresponds to greater benefits.

RMSs are used in several application scenarios, such as online trading and e-commerce frameworks (Tadelis, 2016), service oriented applications (Wahab et al., 2015), peer-to-peer applications (Marti and Garcia-Molina, 2006), collaborative intrusion detection systems (Vasilomanolakis et al., 2015), social networks (Awuor et al., 2018) and crowdsourcing (Wang et al., 2016).

Depending on the specific features of the considered application domain, RMSs can be designed according to a centralized or a decentralized architecture. Centralized RMSs relies on a single server, which collects information about interactions and computes a global and unique reputation value for each agent. Such a model is best suited for those applications where interactions already follow centralized paradigm, such as many e-commerce systems. On the other hand, distributed RMSs, are characterized by the lack of central servers which manage and control the interactions among agents. In such systems, the estimation of the reputation values is performed by using feedbacks provided by the agents involved, and is based on the execution of a distributed cooperation algorithm. Distributed cooperation algorithms, generally allow the whole community to contribute in the reputation evaluation (Hendrikx et al., 2015).

A distributed architecture prevents the existence of a single point of failure and represents a well scalable solution because of the elimination of a potential performance bottleneck. However, such an architectural choice poses several challenges to designers. First of all, evaluating the accuracy of reputation values estimated by the RMS and the time required to convergence is not a trivial task. Some RMSs are based on a sound mathematical formulation, which allows to theoretically evaluate them (Kamvar et al., 2003), but its underlying assumption is not always valid and represents a strong constraint for designing new systems. Moreover, it is often difficult to distinguish the effect of the adopted reputation model from that of the distributed protocol used to spread information over the agent network. Finally, another critical issue is that distributed RMSs are sensitive to fake information disseminated into the system by malicious users. Such a behavior represents an actual security attack, since, by altering reputation estimations produced by RMSs, it is possible to induce users to interact with malicious agents. The level of vulnerability of a specific RMS to different types of attack depends on the design of its components and it is very difficult to evaluate the impact of different choices since the design phase.

A wide range of efforts has been devoted to address these challenges. Nevertheless, no simulation environments which allow to address all these issues through a general approach that can be applied to different application scenarios were found in the literature.

In order to address the above issues, this thesis presents DRESS, a Distributed RMS Evaluation Simulation Software, which allows researchers to evaluate the performance of a generic RMS and its vulnerability against several security attacks. DRESS allows to compare distinct RMSs and different design choices through a set of well-defined metrics, also supporting dynamic and large-scale simulations. DRESS simulates a distributed environment where several agents interact, and allows designers to define the specific features of the RMS to be evaluated, the behavior of each agent, and the set of security attacks to simulate. A set of high-level interfaces allows to disregard some low level details (e.g., implementing the agent communications, driving the simulations), so that the researcher can focus on more important tasks, such as defining new reputation algorithms, or selecting the specific features to produce the desired robustness. DRESS also represents an automatic assessment tool aimed at computing quantitative metrics to evaluate the vulnerability of a RMS to different attacks. These results can be immediately used to show the effects of different design choices on the RMS's performance.

1.2 Contributions

The main contributions of the work presented in this dissertation are:

- The design and the implementation of a Distributed RMS Evaluation Simulation Software, which allows a researcher to evaluate the performance of a generic RMS and its vulnerability against several security attacks. DRESS allows to compare distinct RMSs and different design choices through a set of well-defined metrics, also supporting dynamic and large-scale simulations.

- The provision of a solid formulation of the RMS components, a formal model of four widely diffused security attacks, and a wider set of evaluation metrics.

- The test of all DRESS functionalities through a case study and the description of how a researcher can use this tool for RMS analysis.

- The use of DRESS in a real case to design and implement an energy sharing system based on user behaviors and preferences. Moreover the problem of optimizing the performance of an energy sharing system while considering realistic user behavioral models in terms of preferences and engagement is defined. The problem using MILP and the demonstration that it is NP-Hard is formulated. The proposition of an heuristic based on reinforcement learning to learn the user behavioral model while optimizing the system performance is presented. Finally, the proposed approach is compared with state-of-the-art solutions using simulations based on real traces. Results show that the proposed system significantly outperforms existing approaches by effectively learning the user preference.

1.3 Dissertation Outline

The remainder of the dissertation is organized as follows.

Chapter 2 describes the main features common to most of RMSs presented in the literature, providing the formal model adopted in the simulation framework, and describes the most common attacks on the security of such systems. The same chapter provides a description of the agent-based DRESS architecture, presents the set of functionalities offered by the simulation framework to allow the definition of new RMSs and describes the available evaluation metrics.

Chapter 3 describes a sample case in which DRESS is used to simulate different security attacks to a specific RMS.

Finally, Chapter 4 proposes a case of real use of DRESS in the simulation of energy exchange between residential electricity producers within smartgrid and in the implementation of an RMS for the optimization of this energy exchange system. The use of DRESS has not only allowed to implement an RMS taking into account the preferences of users, identifying the most collaborative, but has also provided significant support for the design of a viable solution for maximizing the amount of energy transferred.

1.4 Publications

Parts of the work in this thesis have been published in several referred conference proceedings and journals:

- Vincenzo Agate, Alessandra De Paola, Giuseppe Lo Re and Marco Morana. DRESS: A Distributed RMS Evaluation Simulation Software. International Journal of Intelligent Information Technologies (IJIIT), 16(3), 2020.

- Vincenzo Agate, Alessandra De Paola, Giuseppe Lo Re, and Marco Morana. A Platform for the Evaluation of Distributed Reputation Algorithms. In 22nd IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications, DSRT 2018, Madrid, Spain, October 15-17, 2018, pages 182-189, October 2018.

- Vincenzo Agate, Alessandra De Paola, Giuseppe Lo Re, and Marco Morana. Vulnerability Evaluation of Distributed Reputation Management Systems. In The 10th EAI International Conference on Performance Evaluation Methodologies, VALUETOOLS2016, pages 235-242, 2017.

- Vincenzo Agate, Alessandra De Paola, Salvatore Gaglio, Giuseppe Lo Re, and Marco Morana. A Framework for Parallel Assessment of Reputation Management Systems. In Proceedings of the 17th International Conference on Computer Systems and Technologies 2016, CompSysTech '16, pages 121-128, New York, NY, USA, 2016. ACM.

- Vincenzo Agate, Alessandra De Paola, Giuseppe Lo Re, and Marco Morana. A Simulation Framework for Evaluating Distributed Reputation Management Systems. In Distributed Computing and Artificial Intelligence, 13th International Conference, pages 247-254, Cham, 2016. Springer International Publishing.

Other articles have been submitted to international conferences and journals, and are accepted or currently under review:

- Vincenzo Agate, Atieh R. Khamesi, Salvatore Gaglio and Simone Silvestri. Enabling peer-to-peer User-Preference-Aware Energy Sharing Through Reinforcement Learning. Submitted to IEEE International Conference on Communications (Accepted).

- Vincenzo Agate, Alessandra De Paola, Giuseppe Lo Re, and Marco Morana. A Simulation Software for the Vulnerability Evaluation of Reputation Management Systems. Submitted to ACM Transactions on Computer Systems.

During the PhD, the following other works were produced:

- Vincenzo Agate, Pierluca Ferraro, and Salvatore Gaglio. A Cognitive Architecture for Ambient Intelligence Systems. In International Workshop on Artificial Intelligence and Cognition (AIC 2018), Palermo, Italy, July 2018.

- Vincenzo Agate, Federico Concone, and Pierluca Ferraro. WiP: Smart Services for an Augmented Campus. In The 4rd IEEE International Conference on Smart Computing (SMARTCOMP 2018), Taormina, Italy, June 2018.

- Vincenzo Agate and Salvatore Gaglio. A Gesture Recognition Framework for Exploring Museum Exhibitions. In International Conference on Advanced Visual Interfaces (AVI 2018), Castiglione della Pescaia, Italy, May 2018.

- Vincenzo Agate, Calogero Crapanzano, Alessandra De Paola, Salvatore Gaglio, and Goffredo La Loggia. SESAMO: An Integrated Framework for Gathering, Managing and Sharing Environmental Data. In Proceedings of the 17th International Conference on Computer Systems and Technologies 2016, CompSysTech '16, pages 137-144, New York, NY, USA, 2016. ACM.

- Tiziana Catarci, Francesco Leotta, Andrea Marrella, Massimo Mecella, Daniele Sora, Pietro Cottone, Giuseppe Lo Re, Marco Morana, Marco Ortolani, Vincenzo Agate, Giovanni Renato Meschino, Giovanni Pecoraro, and Gabriele Pergola. Your Friends Mention It. What About Visiting It?: A Mobile Social-Based Sightseeing Application. In Proceedings of the International Working Conference on Advanced Visual Interfaces, AVI 2016, Bari, Italy, June 7-10, 2016, pages 300-301, 2016.

Chapter 2

Model Assumptions and System Architecture

In distributed environments, a number of unknown entities interact with each other in order to achieve complex goals. In such a situation, these entities could adopt selfish behavior so as to increase their own advantages. For this reason, intelligent techniques for estimating agents' reputations are required. Reputation Management Systems exploit feedbacks provided by the agents themselves, and are used to predict their future behaviors. Unfortunately, these systems are very sensitive to fake information injected into the system by malicious users; thus, evaluating the robustness of a Reputation Management System to malicious behavior is a challenging task. This chapter presents DRESS, an agent-based simulation software designed to support researchers in the evaluation of distributed Reputation Management Systems since the design phase. First, the literature related to simulators for RMS is reviewed. Second, a formal model for the representation of reputation management systems and a set of the most effective RMS attacks are defined. Then, the general architecture of the proposed system is outlined, highlighting the aspects that ensure its adaptivity and independence of the context. Furthermore, an unbiased measure of the error between the reputation estimated by the RMS and the true value of agent's cooperativeness are described. Finally, some default vulnerability metrics are introduced, in order to measure the robustness of a RMS against the security attacks.

2.1 Existing Approaches

One of the earliest simulators for reputation systems is the ART-testbed (*Agent Reputation and Trust*) (Fullam et al., 2005), whose main goal is the comparison of two RMSs through objective metrics. It allows to evaluate different strategies in terms of the utility obtained by each agent, and also to estimate the accuracy of the reputation with respect to the ground truth. However, ART does not support any specific analysis of the vulnerabilities brought by the distributed RMS protocol, that instead is fully enabled by DRESS. Moreover, ART has been designed for Multi-Agent Systems (MAS) and its main limitation is that it forces the user to meet specific constraints while modeling the RMS or the application scenario.

This limitation is common to many other simulation tools proposed in the literature. TREET (Kerr and Cohen, 2010) was designed to evaluate RMSs in a marketplace scenario by measuring their robustness to some attacks (e.g., reputation lag, proliferation, and value imbalance). This testbed introduces a certain degree of dynamism by allowing agents to randomly join or leave the simulation, but such events cannot be scheduled in advance. The testbed proposed in (Chandrasekaran and Esfandiari, 2015) requires the RMS to be modeled as a sequence of graphs transformations and allows to evaluate the RMS vulnerability to few attacks (e.g., slandering and self-promotion). However, the adoption of such a very specific model may represent a limitation for many RMS designers and the framework does not allow to simulate dynamic agents behaviors.

DART (Salehi-Abari and White, 2012) adopts a model based on prisoner's dilemma games to analyze different RMSs, even against some security attacks. Its main limitation is that the adopted decision making mechanism highly influences the experimental evaluation, so preventing for a clear and unbiased assessment of other RMS's components.

Allowing the designers to implement their own model of RMS, and making it easily comparable with other solutions drove some works such as, ATB (Jelenc et al., 2013), TRMSim-WSN (Mármol and Pérez, 2009), and QTM (West et al., 2010). Nevertheless, ATB mainly focuses on the decision making mechanism and neglects other relevant components of a RMS, limiting once again the range of systems that can be evaluated. An analogous limitation characterizes TRMSim-

WSN, which is intended only to model RMSs over wireless sensor networks. On the other hand, QTM supports the definition of new RMSs without imposing constraints on the application scenario or the RMS model, and presents also a high flexibility regarding the personalization of the security attacks. Researchers can simulate an attack by using one of the six user models provided by the testbed, or by defining new more complex user models. QTM adopts the *hit rate*, i.e., the percentage of successful transactions performed by cooperative users, as the sole metric to evaluate the RMS's performance. Thus, the set of experiments that can be run is quite limited and it is not possible to deeply evaluate which kind of vulnerability mainly affects the considered system. The testbed proposed in (Irissappane et al., 2012) introduces a wider set of metrics, even though it allows to model as malicious only those agents that provide unfair ratings, so neglecting any aspect related to the cooperativeness degree as service provider. TOSim (Zhang et al., 2007) addresses the need for a flexible simulation by proposing a modular framework, also intended for performing scalable simulations, whose application is however limited to the evaluation of P2P overlay systems.

With respect to other works presented in the literature, DRESS is characterized by the following desirable characteristics, partially identified in (Koutrouli and Tsalgatidou, 2016):

- the inclusion of an abstract RMS model, that can be easily implemented to represent a specific system, and that maintains the independence from a specific application scenario;
- the inclusion of formal models for most common security attacks against RMSs;
- the possibility of evaluating the effects of the decentralized nature of the RMS under evaluation, by comparing the obtained reputation estimation with reputation values obtainable by a centralized RMS which knows the actual outcome of all transactions;
- the availability of a set of well-defined metrics, capable of evaluating both the RMS's accuracy and its vulnerability to security attacks while considering different facets;

- a high flexibility in varying the set of security attacks, the aspects of the RMS to be analyzed, and the simulation scenarios;
- the possibility of performing large-scale simulations.

2.2 Reputation Management Systems

Regardless of the specific application domain, Reputation Management Systems share the following common features. They are based on feedbacks provided by the agents involved in interactions and use such information to build an overall estimation of agents' reputation. Feedbacks may be provided by human users, as in e-commerce frameworks, or by software agents, as in QoS-based Service Oriented Architectures, and this difference does not relevantly affect the formal definition of RMS's algorithms. For the sake of generality, in the rest of this dissertation the term *agent* is adopted to indicate the generic participant of the RMS, both human user and software agent. Reputation values are used to take future decisions, since they enable a prediction of the future behavior of agents. Such decisions may be taken directly by users, that may choose to not interact with users characterized by a low reputation, or by software agents that may apply specific optimization policies.

Given the above considerations, in what follows it is presented the adopted RMS model which allows to highlight the main components common to most of the RMSs described in the literature. Such a model is also exploited to provide a formal definition of some security attacks which can affect the performance of such systems.

2.2.1 RMS Model

Let $V(t)$ be the set of agents, also called nodes, involved in the RMS, at time t. Each agent $v_i \in V(t)$ acts as resource provider and can be characterized by an intrinsic level of cooperativeness, $\tilde{R}_i$. This information is unknown to other agents and its estimation is one of the main goal of the RMS. Cooperativeness can be related to quantity or quality of the provided resources, depending on the specific distributed application. The cooperativeness can vary over time, thus it has to be indicated as $\tilde{R}_i(t)$.

In a centralized RMS, after each interaction, the consumer agent v_i sends a feedback $f_{ij}(t)$ about the provider agent v_j to the centralized server, which aggregates feedbacks to produce the reputation value of the provider, i.e., $R_j(t)$. The centralized information fusion function Φ_C can accept as input the whole history of collected feedbacks, as expressed by the following equation:

$$R_j(t) = \Phi_C\left([f_{ij}(t')]_{\substack{t'=0:t-1\\ v_i\in V(t), v_i\neq v_j}}\right). \tag{2.1}$$

In distributed RMSs, on the contrary, there is not a centralized server responsible for collecting feedbacks, and the agents are directly responsible for guaranteeing that reputation information flows through the network. The interactions among agents, aimed to exchange both services and reputation information, may be represented through an overlay *reputation network*, whose structure can be static or can varies over time. At time t, the reputation network is represented by a set of vertices and edges, i.e., $G(t) = (V(t), E(t))$. A network vertex in $V(t)$ represents an agent which is active at time t, while the presence of an edge $(v_i, v_j) \in E(t)$ indicates that the agents v_i and v_j know each other at time t. In such systems, each agent v_i estimates its own reputation of agent v_j, i.e., $r_{ij}(t)$, and sends its opinions to its neighborhood in the reputation network, according to a distributed *gossip protocol*. Messages collected from other nodes are used, together with the past experience, in order to update the reputation estimation.

If neglecting the messages received from other agents, each agent v_i can even only exploit direct feedbacks in order to build a *local reputation* value, $l_{ij}(t)$:

$$l_{ij}(t) = \Lambda\left([f_{ij}(t')]_{t'=0:t}\right), \tag{2.2}$$

where Λ is the *local reputation* function, which considers as input the whole history of direct feedbacks about agent v_j and computes its current local reputation value.

On the other hand, according to the *gossip protocol* adopted by the distributed RMS, at each time step, each agent v_i may receive a set of messages from its neighborhood in the reputation network, i.e., $\mathcal{M}_i(t)$. If $N_i(t)$ is defined as the set of neighbors of v_i at time t, i.e., $N_i(t) = \{v_k \in V(t) : \exists(v_i, v_k) \in E(t)\}$, $\mathcal{M}_i(t)$ can be defined as follows:

$$\mathcal{M}_i(t) = \{\mathcal{M}_{k\to i}(t), \forall v_k \in N_i(t)\}\,, \tag{2.3}$$

where $\mathcal{M}_{k\to i}(t)$ is the message received by v_i from v_k at time step t, or the void message if no communication occurs between these two agents.

Messages may contain information about every agent v_j regarding which the sending agent v_i has an opinion. We define $O_i(t)$ as the set of agents for which the agent v_i has an opinion at time t. In general, the message generation, performed through the function Γ can consider the whole history of direct feedbacks and received information, as specified by the following equation:

$$\mathcal{M}_{i\to k}(t) = \Gamma\left([f_{ij}(t')]_{\substack{t'=0:t-1\\ v_j\in O_i(t-1)}}, [\mathcal{M}_i(t')]_{t'=0:t-1}\right). \tag{2.4}$$

Messages collected from other agents, together with local reputation, can allow agent v_i to estimate the reputation of agent v_j, as follows:

$$r_{ij}(t) = \Phi\left([l_{ij}(t')]_{t'=0:t}, [\mathcal{M}_i(t')]_{t'=0:t}\right), \tag{2.5}$$

where Φ is the *information fusion* function which can accept as input the whole history of collected feedbacks, expressed by the local reputation, and of received messages.

It is worth noticing that RMSs presented in the literature do not consider the whole history of feedbacks and messages at each time step. On the contrary, commonly, current reputation depends on last reputation values and on latest local reputation and messages, as follows:

$$r_{ij}(t) = \Phi\left(r_{ij}(t-1), l_{ij}(t), \mathcal{M}_i(t)\right). \tag{2.6}$$

2.2.2 Security Attacks on RMSs

This subsection introduces the most common attacks capable of weakening the correct functioning of distributed RMSs. Since the correct functioning of distributed

RMSs depends on the voluntary participation of agents in the coordinated effort devoted to reputation estimation, fake feedbacks represent one of the most serious threats. Indeed, while honest agents propagate truthful information, a malicious agent may propagate fake information, in order to alter the reputation value for some agents and achieve its own goals, that may contrast with systems goals.

Several type of security attacks which exploit the distributed nature of RMSs and their dependence on agent feedbacks are reported in the literature (Hoffman et al., 2009; Sun and Liu, 2012). Malicious agents performing these attacks are *insiders*, i.e., authorized agents of the system that legitimately participate in reputation evaluation. According to this assumption, the work in this thesis does not consider security issues related to attacks performed by outsiders, such as threats to authentication, integrity and confidentiality. Several approaches presented in the literature can be adopted in order to guarantee a correct membership management, such as that proposed in (Johansen et al., 2015). The following analysis also assumes that attackers can obtain multiple identities and that are able to cooperate in order to perform an orchestrated attack. A classification of security attacks to RMSs performed by insiders can be made on the basis of their goals: *promoting*, *slandering*, *whitewashing*, and *traitor* attacks. *Promoting* and *slandering* attacks are characterized by the diffusion of fake information into the reputation network, while *whitewashing* and *traitor* attacks are characterized by agents that adapt its own behavior in order avoid the effect of an accurate estimation of their reputation.

Promoting Attack

The goal of *promoting* attacks (Lian et al., 2007) is to increase the reputation of a *target* agent, in order to hide its antisocial behavior or to achieve an unjustified advantage over its competitors. For example, in an e-commerce scenario, a malicious agent may want to increase its own reputation in order to hide its actual, unworthy, behavior. Promoting attacks are usually performed by introducing fake positive information into the reputation network, through the gossip protocol. Such an attack is generally performed through an orchestrated plan which involves several malicious agents, in order to bypass the typical rule which prevents an agent from spreading feedbacks and information about itself. For the sake of our analyses, it is possible to neglect whether an orchestrated attack is performed by distinct malicious agents or by multiple identities of a single malicious agent.

Formally, if v_i is the target agent which takes advantage of the attack, and V^* is the set of malicious agents, for each $v_j \in V^*$, performing a promoting attack corresponds to send to each neighbor v_k the message $\mathcal{M}^*$ which maximizes the reputation that v_k holds about the target agent v_i. That corresponds to solve the following problem:

$$\forall v_k \in N_j(t), \quad \mathcal{M}_{j\to k}(t) = \underset{\mathcal{M}^*}{\operatorname{argmax}} \left(r_{ki}(t)\right) = \underset{\mathcal{M}^*}{\operatorname{argmax}} \left(\Phi\left(r_{ki}(t-1), l_{ki}(t), \mathcal{M}^*\right)\right). \tag{2.7}$$

It is worth noticing, that v_j performs this optimization without knowing r_{ki} $(t-1)$ and $l_{ki}(t)$, which are known only to destination agent v_k.

From eq. 2.7, it derives that the RMS component capable of withstanding promoting attacks is the information fusion function, i.e., Φ, which determines the balance among past history, direct experience, and gossiped information. In order to increase the RMS resistance to fake information, the information fusion function could also include different weights to obtained information, e.g., on the basis of reputation of gossiper agents.

Slandering Attack

The goal of *slandering* attacks (Ba and Pavlou, 2002) is to decrease the reputation of some *victim* agents. Differently from promoting attacks, slandering attacks can be performed by a single malicious agent. Nevertheless, in large communities, an isolated intervention would have a limited effect, thus also slandering attacks are generally performed through an orchestrated action which involves several malicious agents. Such an attack is typical of e-commerce systems, where the attacker aims to sabotage a competitor in order to obtain an indirect economic profit. The slandering attack is performed by introducing fake negative information through the gossip protocol.

If v_i is the victim agent, and V^* is the set of malicious malicious agents, each $v_j \in V^*$ performs the *slandering* attack by sending to its neighbors the message $\mathcal{M}^*$

which allows to minimize their reputation about v_i, as expressed by the following equation:

$$\forall v_k \in N_j(t), \quad \mathcal{M}_{j\to k}(t) = \underset{\mathcal{M}^*}{\operatorname{argmin}}\left(r_{ki}(t)\right) = \underset{\mathcal{M}^*}{\operatorname{argmin}}\left(\Phi\left(r_{ki}(t-1), l_{ki}(t), \mathcal{M}^*\right)\right). \tag{2.8}$$

Similarly as the promoting attack, the information fusion function is the main component that can let the system resist through an opportune balancing of direct experience and gossiped information.

Whitewashing Attack

A malicious agent performing a *whitewashing* attack (Feldman et al., 2004) aims to avoid the consequences of its selfish past behavior, by leaving the system and rejoining it with a new identity, thus to obtain the default reputation value assigned to new users. The main vulnerability exploited by this type of attack is the choice of a default reputation value which is comparable with the long-term reputation of cooperative agents. Such a choice brings to RMSs which adopt an initial optimistic approach and rely on negative feedbacks in order to discover malicious behaviors.

Let r_0 be the default reputation value assigned to new agents, i.e., $r_{ij}(0) = r_0, \forall v_i, v_j \in V(0)$. The whitewashing attack can be modeled as the introduction of a new node v_k in the reputation network in a given time step t:

$$v_k \notin V(t-1), v_k \in V(t). \tag{2.9}$$

Agents that will interact with the new agent v_k at time t, will assign it the default reputation value:

$$r_{jk}(t) = r_0, \quad \forall v_j \in H_k(t), \tag{2.10}$$

where $H_k(t)$ is the set of agents which hold an opinion of agent v_k at time t, i.e. $H_k(t) = \{v_j : v_k \in O_j(t)\}$.

If we define u_i as the utility function which models the dependence of benefits perceived by agent v_i on costs due to maintain its cooperativeness and on its reputation estimated by other agents, and c_{new} as the cost of creating of a new identity, the whitewashing attack is advantageous for the agent v_i if:

$$u_i\left(\tilde{R}_i(t-1), [r_{ji}(t-1)]_{v_j \in H_i(t-1)}\right) < u_k\left(\tilde{R}_k(t), [r_{jk}(t)]_{v_j \in H_k(t)}\right) - c_{\text{new}}. \quad (2.11)$$

On the basis of such analysis, we can state that a greater resistance is expected by RMSs that impose a low initial reputation value r_0 and use positive feedbacks to rise the reputation value.

Traitor Attack

In such an attack, a *traitor* agent (Marti and Garcia-Molina, 2006), alternates cooperative and selfish behavior in order to maintain a reasonable reputation value while keeps abusing system resources. Thus, a traitor acts honestly for a limited portion of time in order to increase its reputation. Once such goal is achieved, it starts to abuse system resources, and maintains an antisocial behavior until its reputation became too low.

If ΔT_1, ΔT_2 and ΔT_3 are consecutive time intervals, the behavior of a traitor v_i can be represented as follows:

$$\begin{cases} \tilde{R}_i(t_1) > \tilde{R}_i(t_2) & \forall t_1 \in \Delta T_1, t_2 \in \Delta T_2; \\ \tilde{R}_i(t_2) < \tilde{R}_i(t_3) & \forall t_2 \in \Delta T_2, t_3 \in \Delta T_3. \end{cases} \quad (2.12)$$

The length of cooperative and non cooperative time intervals is generally selected in order to guarantee that the utility perceived by the traitor never goes under a given threshold, i.e., u_{th}. Thus, from the point of view of the traitor, the selection of its cooperativeness has to respect the following constraint:

$$u_i\left(\tilde{R}_i(t), [r_{ji}(t)]_{v_j \in H_i(t)}\right) > u_{th}, \ \forall t \in \Delta T_2. \quad (2.13)$$

RMSs more vulnerable to such type of attack are those which weight the past history more than the recent experience.

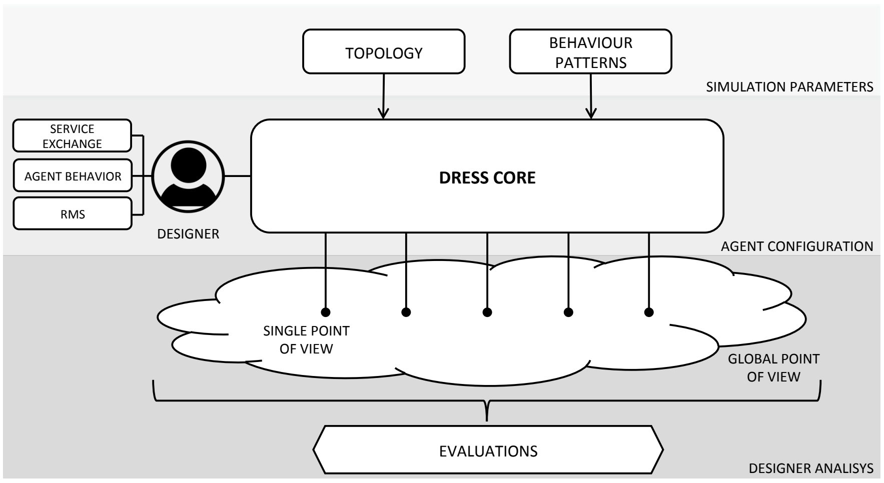


Figure 2.1: Overview of the DRESS architecture.

2.3 Simulation Models

DRESS aims to provide designers with high-level interfaces that allow to easily define the behavior of a specific RMS according to the above described formal model and to simply configure a simulation scenario, while neglecting low-level implementation details[1]. A preliminary overview of the DRESS architecture is shown in Fig. 2.1.

To achieve this goal, DRESS adopts a two-level architecture, where the upper level is designed for modeling the RMS and the simulation scenario, and the lower level implements the communication and control primitives needed for actually driving the simulation. At the topmost layer, the RMS is modeled as a fully distributed system in which autonomous agents interact in order to exchange services. Such a behavior is implemented in the lowest layer through a computer cluster where each agent is mapped on a different process running on some physical node, and where processes communicate with each other by exchanging messages by means of the Message Exchange paradigm and using Message Passing Interface (MPI). The MPI library allows each process to send and receive messages to and from any other process within the simulation environment.

[1]The "DRESS Reference Manual" is available at `http://diid.unipa.it/networks/ndslab/DRESS/refman.pdf`

The interaction between designers and DRESS involves only the highest level, whilst all the low level functionalities remain hidden.

The simulator structure and its working mechanisms are developed with the aim of hosting a generic RMS; thus, designers can define the behavior of a specific RMS by implementing a set of specific classes which inherit their structure from the abstract classes included in the DRESS library.

According to the available interfaces, designers can re-define some system features in order to model a novel distributed application and its RMS, or they can combine existing solutions already available in DRESS.

In particular it is possible to define the logic behind the agent service exchange mechanism, the characteristics of a specific RMS through the implementation of its algorithms and its communication patterns, and, finally, the set of different behaviors of agents to be simulated, both considering the collaboration degree adopted in service exchanging and the truth exhibited while participating to RMS functioning.

Even if these tasks require that designers are endowed with some programming skills, they dramatically expand the range of applications and RMSs that is possible to evaluate.

Once the distributed application and the RMS are fully defined, it is possible to evaluate the RMS's performance in different simulation scenarios. The first information required to define a simulation scenario is the topology of the distributed RMS, which specifies the set of neighbors each agent can interact with. The simulation scenario is completed by defining the behavior pattern for each agent over time. DRESS contains the definition of some basic behaviors, such as cooperative/selfish, honest/slanderer/promoter, which can be applied with different degrees and changing over time. These behaviors can also be extended by including in the simulation the new behaviors defined by the developer.

At the end of the simulation, DRESS allows to analyze the RMS's performance from the point of view of a specific agent, or by obtaining a global evaluation. Focusing on a single point of view it is possible to analyze the trend over time of the reputation of a given agent as estimated by another one, and of the percentage of satisfied service requests for a specific agent.

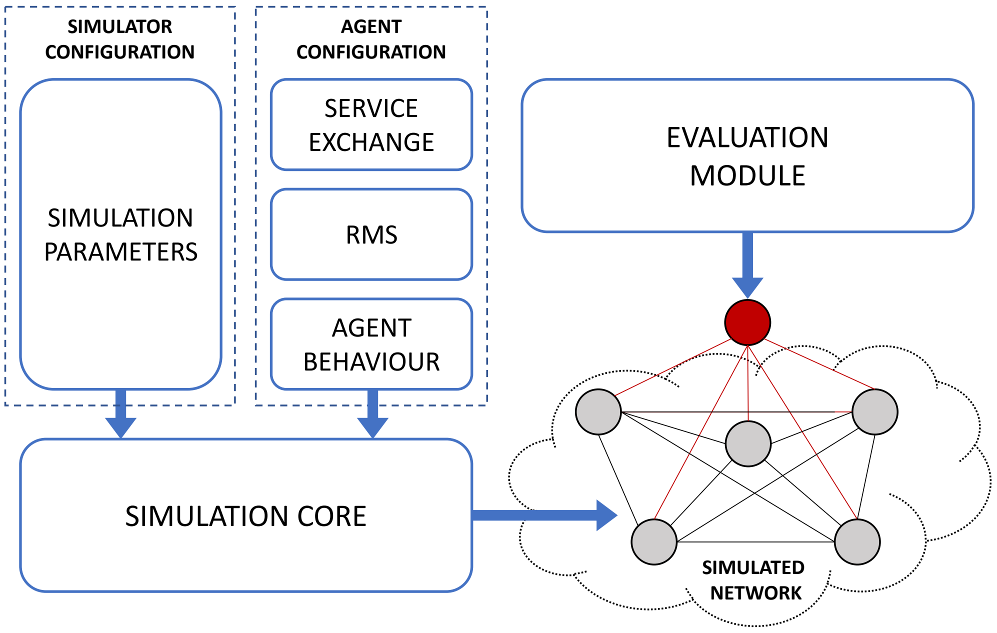


Figure 2.2: Logical overview of the simulation platform. Users can specify simulation parameters, agent characteristics, and algorithms used by the RMS. The simulated network is analyzed through a customizable evaluation module.

From a global point of view, DRESS allows to compare the performance of the RMS under evaluation with that of an ideal centralized RMS which is not affected by fake information and by the bias introduced by the gossip protocol, thus to obtain a global error index. Moreover, it provides the quantitative evaluation of the RMS's vulnerability to the considered security attacks, according to some well-defined metrics.

2.3.1 Platform Architecture

As shown in in Fig. 2.2, the simulator can be adapted to model different RMSs by specifying the *agent configuration*. This phase allows to detail the service request/response policies, the algorithms behind the RMS, and the different behaviors that each agent can follow over time. Once such configuration is completed, it is possible to perform the *simulation configuration* by specifying a set of available parameters. In this phase it's possible to specify the topology of the reputation network that lists the set of neighbours each agent can interact with. Moreover, users can specify the behavior each agent has to follow during the simulation, and the possible sequence of actions to perform. Finally, the *Evaluation Module*, which analyzes the RMSs performance at the end of the simulation, can be customized by defining any ad-hoc evaluation metrics.

During the simulation, the agents interact with each other in order to provide/receive services and obtain a reward according to the synchronous, time-discrete model. The role of the RMS is to apply an incentive mechanism that makes the reward proportional to the cooperativeness of each agent, computed by taking into account the whole community of agents. As result, agents are able to select a service provider according to the policy established by the RMS, e.g., by selecting the provider which corresponds to the highest expected utility.

The simulation evolves through a set of rounds during which all the agents cyclically perform the same sequence of steps. The behavior of a single agent within a round is defined by implementing some functions inherited from an abstract *agent class* provided with the simulation library. More specifically, the designers can specify:

- the service exchange logic, which rules the sending of service requests and replies (according to the RMS policy);
- the set of RMS algorithms, which specifies how to spread agents' opinion to their neighbourhood and how to compute the reputation values;
- the agent behavior, which models its cooperativeness during the service exchange.

The adoption of the *composition over inheritance* principle allows to easily define new agents whose behavior is a combination of two behavioral dimensions: those identified as service providers and those specified as members of the RMS. Moreover, new behaviors can also be defined by inheriting the available abstract classes.

The *Evaluation Module* allows to analyze the RMS's performance with different levels of detail. For instance, the simulator allows to evaluate the average reputation estimated by the RMS over the whole network, but it makes also possible to analyze with a high degree of granularity the trend over time of the reputation of

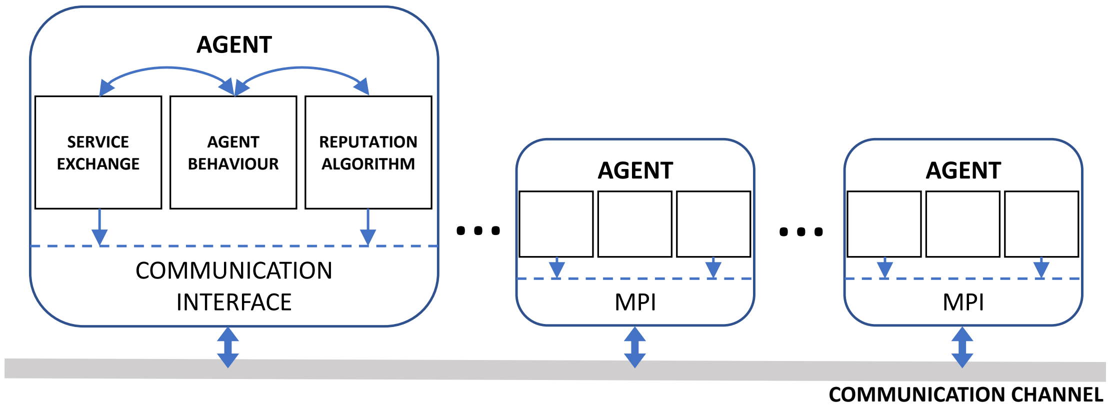


Figure 2.3: The distributed multi-agent simulation scheme. Agents are mapped to system processes that communicate through the MPI protocol.

a given agent as estimated by one of its neighbors. Furthermore, the *Evaluation Module* allows to compare such outcomes with those obtained by the *truth-holder* in order to understand whether the errors are due to the reputation algorithm, or to possible bias introduced by false user feedbacks.

2.3.2 Simulation Environment

DRESS aims to support large-scale simulations modeling real-world scenarios.

One of the most important requirements of a simulation environment for RMS evaluation is to guarantee a high degree of parallelism. To meet this constraint, DRESS takes advantage of a distributed environment where each agent is simulated by a process running on a computer cluster. Fig. 2.3 highlights the four elements characterizing an agent. Besides the aforementioned components, i.e., *service exchange*, *RMS algorithms*, and *agent behavior*, a *communication interface* is needed to enable messaging over a communication channel. In order to provide the programmers with a standard protocol, communication between processes is based on Message Passing Interface.

The core of the platform is written in C++ and exploits MPICH, a MPI library available for many UNIX like distributions and Windows OS, to provide the designer with an easy tool to implement his own distributed algorithm.

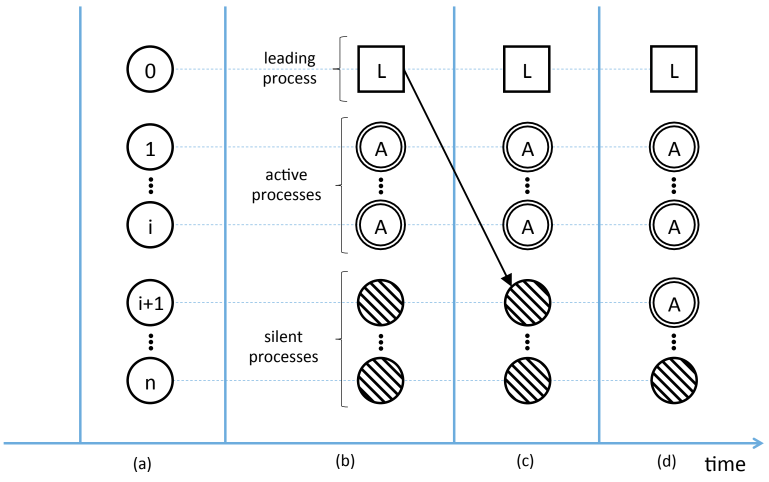


Figure 2.4: Pool of processes managed by DRESS: (a) the MPI environment starts a set of identical processes; (b) by reading the simulation parameters, each process adopts its own behavior among *leading*, *active*, or *silent*. (c) MPI messages sent by the leading process cause some other processes to change their status (d).

According to the MPI programming model, the distributed execution is obtained by starting a fixed number of processes executing the same program, as depicted in Fig. 2.4-a. The MPI library allows each process to send and receive messages to and from any other process within the simulation environment. The polymorphic behavior of various processes is obtained through a set of input parameters which are used to drive the execution flow.

In order to model a dynamic network, the fixed set of processes started at the simulation start includes both a group of processes representing active agents, and a group of silent processes corresponding to those agents that will successively join the RMS network. While all processes are started as identical copies of the same initial process, the choice of the actual process behavior depends on the simulation parameters, as depicted in Fig. 2.4-b.

The mapping of each agent in a process allows DRESS to flexibly adapt the simulation to available computation resources and leads to a software which elegantly mirrors the distributed architecture of RMSs.

The set of processes also includes a leading process responsible for managing the agents' life cycle. When it is required to simulate a new agent joining the RMS network, the leading process wakes up one of the silent processes by sending

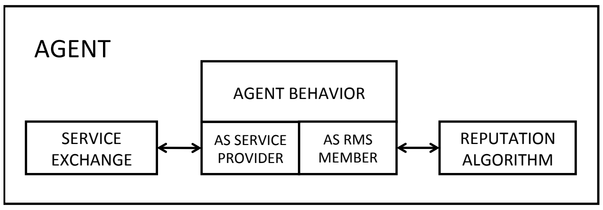


Figure 2.5: Components of a specific agent.

an opportune MPI message (see Fig. 2.4-c). As consequence, at the next time interval, this process is added to the pool of processes which model active agents (see Fig. 2.4-d). Analogously, when the simulated sequence of events includes the exit of an agent from the RMS system, the leading process switches the status of the corresponding process to silent.

It is worth noticing that the centralized coordination performed by the leading process regards only the evolution of the state of processes, while service exchanging and communications within the RMS are performed in a totally distributed way.

During the simulation, a number of *log* files containing information on the tasks performed by every agent are created. Using such files, the analyst may choose to collect data about the reputation values computed, or observed, by the agents, the number of incoming/outcoming service requests for each node of the network, the number of satisfied/unsatisfied requests, and the cooperativeness degree of each agent.

2.3.3 Agent-based Model

Each agent participating to the RMS interacts with other agents to receive some services, and consequently perceives a certain utility. Generally, a RMS applies some incentive mechanisms which make such utility proportional to the cooperativeness of the agent, i.e., $\tilde{R}_i(t)$. Distributed RMSs estimate this cooperativeness degree by relying on the whole community of agents, which further interact by exchanging information about their past experiences and opinions. As result, agents are able to select a service provider according to the policy established by the

RMS, e.g., by selecting the provider which corresponds to the highest expected utility.

In the proposed simulation software, agent interactions are orchestrated according to the synchronous time-discrete model proposed in (Lynch, 1996). The simulation is composed by a set of rounds, during which all the agents cyclically perform the same sequence of steps, such as sending service requests to other agents, replying to received requests, and communicating with neighbor agents to implement the distributed RMS.

The simulation loop is designed with the aim of managing a set of agents, whose structure and methods are defined by an abstract class included in the DRESS library. An actual agent is defined through a specific class which inherits and implements from the abstract class the functions associated to all the steps of a round, as showed in Fig. 2.5, and described in the following.

Service Exchange Model

Without losing generality, interactions between agents are modeled according to a producer-consumer pattern. In a typical scenario, agents can act both as service providers and service consumers, i.e., an agent provides a service to some consumers, while also consuming services from other agents of the network.

Each agent maintains a list of known providers and advertises the exported services by means of the *Service Interface.* This list is shared as specified by the *service announcement* protocol which allows providers to inform some consumers about the set of available services.

The policy for selecting a service provider, is specified by the *service selection* method.

Once a provider has been selected, each consumer can send the service request. During the same simulation step, provider agents receive the requests sent by the consumers and reply to each request according to the behavior defined in the *service reply* method.

It is worth noticing that the *service reply* method is the core of the RMS since it allows to implement the *incentive mechanism* of the modeled RMS, which could discourage antisocial behaviors by rewarding trustworthy agents and limiting ma-

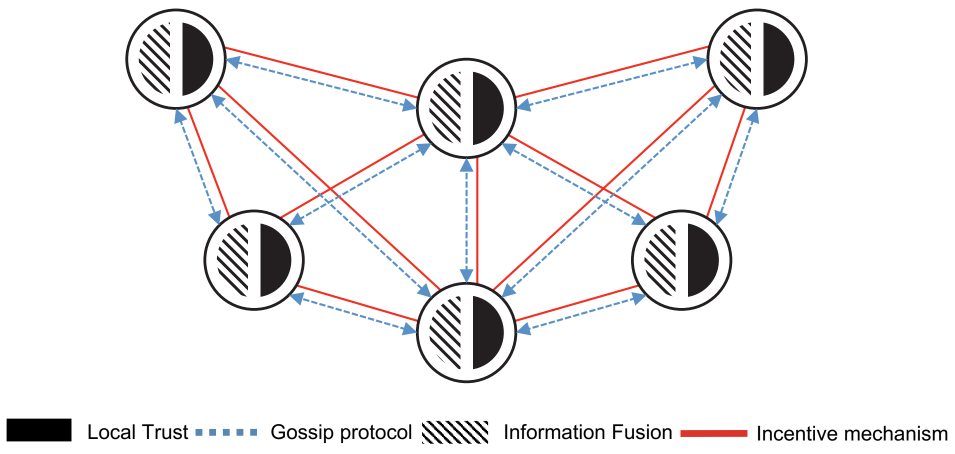


Figure 2.6: The components of a distributed RMS. Each agent privately performs the local trust evaluation and the information fusion algorithms; the gossip protocol and the incentive mechanism regulate the interactions with other agents.

licious ones. E.g, it is possible to define a policy according to which a provider replies only to agents whose reputation is above a given threshold, or randomly decides to reply with a probability proportional to consumer's reputation. Decisions taken by the service reply method, are further influenced by the agent's *behaviour as service provider*, which depends on its cooperativeness, i.e. $\tilde{R}_i(t)$.

Finally, a consumer agent rates each interaction by providing a feedback $f_{ij}(t)$ which is locally stored and further spread over the community according to the reputation algorithm, as detailed in the following.

Reputation Algorithm

One of the main feature of DRESS is the possibility of defining the RMS algorithm, which specifies how to spread agents' opinion to their neighborhood and how to estimate the reputation values by exploiting the available information.

Reputation can be represented as scalar, discrete value (e.g., *good*, *medium*, *bad*), vector containing different parameters (e.g., type of behavior, cooperativeness, level of engagement), or generic object.

Regardless of the specific reputation representation and the reputation management algorithm, four common components were identified, as shown in Fig. 2.6, included in any RMS (Agate et al., 2016a) which implement main functions of the formal model described in Section 2.2.1:

- *Local trust evaluation*;
- *Gossip protocol*;
- *Information fusion*;
- *Incentive mechanism.*

The *local trust evaluation* is the general mechanism an agent uses to make a first estimation of the reputation of other agents. It implements the Λ function and exploits the set of feedbacks f_{ij} stored after each interaction. This method uses only direct interaction outcomes, thus, it can not be used to predict the behavior of previously unseen entities. For this reason, the *local trust evaluation* is supported by two more components, namely the *gossip protocol* and the *information fusion* mechanism.

The *gossip protocol* method implements the Γ function and allows to define which information must be shared with other agents. For instance, a common choice is to share recent reputation values, i.e. $r_{ij}(t)$, with neighbors only.

The *information fusion* method, which implements the Φ function, allows each agent to estimate the reputation of other agents by merging the local trust l_{ij} with gossiped information $\mathcal{M}_i$, provided by the *local trust evaluation* mechanism and the *gossip protocol* respectively. If one of these two parameters is missing, the reputation is computed according to the other. If both values are missing, the information fusion method simply returns the default reputation value. The reputation values computed within this method are updated at each time step.

The *Incentive mechanism* exploits reputation values in order to discourage selfish behaviors and is generally implemented through the service exchange mechanism, as described before.

Agent Behavior

The last characteristic that designers can define is the agent behavior, which models its degree of cooperativeness during the service exchange, and his probity as member of the distributed RMS. With respect to service provisioning, the agent behavior can be fully *cooperative*, fully *selfish*, or partially cooperative, with a degree that can be specified as parameter. As regards the contribution to the RMS, the agent behavior can be *honest*, *slander* (if fake negative feedbacks are provided) or *promoter* (in case of fake positive feedbacks). Moreover the designers can define new behaviors.

Through the definition of the behaviors of different agents, DRESS allows to simulate different types of security attacks that can be classified in two main categories. The former includes attacks in which malicious agents try to alter the RMS performances by providing fake reputation values, the latter involves attackers that operate as bad service providers. *Promoting* and *slandering* are two examples of attacks that require a malicious behavior of a group of agents *as RMS members*, while *whitewashing* and *traitor* attacks imply the malicious behavior of an agent *as service provider* (see Fig. 2.5).

The behavior of an agent *as RMS members* can be changed by altering the content of data sent by means of the gossip protocol, according to equations 2.7 and 2.8, while the behavior *as service provider* can be defined by modifying the cooperativeness parameter the which drives *service reply* method, according to equations 2.11 and 2.13.

In order to allow for a high degree of personalization of the simulation environment, DRESS includes a set of behavioral patterns the user can combine to describe the behavior of specific agents. While configuring a specific simulation scenario, it is possible to set how many agents should exhibit a behavior b, where b can be either atomic, e.g., *slander*, or obtained by composing n atomic behaviors, $b = [b_1, ..., b_n]$. For example, when planning a complex collusion attack, it might be desirable that an agent A contributes both to rise the reputation of an agent, and to reduce that of another one. In such case, its behavior could be expressed as $b_A = [b_{slander}, b_{promoter}]$ to let a specific agent perform both a slandering and a promoting attack, with different targets. Moreover, it is possible to define a behavior which varies over time, simply specifying the time interval during which each pattern has to be followed.

2.4 Evaluation Metrics

This section presents the metrics considered most relevant for a synthetic assessment and under two interesting aspects for researchers: the accuracy of RMS in estimating the reputations of agents and the vulnerability to security attacks in order to determine its robustness.

2.4.1 Accuracy Metrics

In order to evaluate the RMS performance, the simulation platform provides the designers with an unbiased measure of the error between the reputation estimated by the RMS, i.e., $R_i(t)$, and the true value of agent's cooperativeness, i.e., $\tilde{R}_i(t)$. Moreover, in order to highlight the effect of the distributed nature of a RMS on its performance, DRESS evaluates the error with respect to an ideal RMS which adopts the same reputation model, but uses the truthful knowledge of interaction outcomes instead of agents' feedbacks.

In order to define proper accuracy metrics, it is necessary to distinguish between absolute and relative reputation indices. In RMSs which adopt an absolute reputation index, the reputation value of a specific agent is independent of others' reputation. On the contrary, in reputation systems that adopt relative indices, the reputation value is obtained through a normalization process which depends on the reputation values of all known agents. The formulation proposed here assumes an absolute reputation index.

We can assume that reputation and cooperativeness are values in the same domain; we define the *absolute error* as the difference between the cooperativeness of an agent and its reputation. In a centralized RMS, the absolute error obtained about agent v_i is simply represented by the following equation:

$$E_i(t) = |\tilde{R}_i(t) - R_i(t)|. \tag{2.14}$$

On the contrary, in a distributed RMS, where each agent can have a different perception of reputation of agent v_i, it is convenient to consider the error between the *average* reputation and the true cooperativeness. If the *average reputation* of agent v_i at time t is defined as follows:

$$r_i(t) = \frac{\sum\limits_{v_j \in H_i(t)} r_{ji}(t)}{\#\left\{H_i(t)\right\}}, \tag{2.15}$$

where $\#\left\{H_i(t)\right\}$ indicates the number of agents which hold an opinion about the agent v_i at time t. Then, the absolute error on a single agent v_i can be defined according to the following equation:

$$e_i(t) = |\tilde{R}_i(t) - r_i(t)|. \tag{2.16}$$

The absolute error depends on (i) fake feedbacks introduced into the system by malicious agents, (ii) the *gossip protocol*, and (iii) the estimation reputation algorithm, expressed by *local reputation* and *information fusion* functions. In order to enable DRESS to evaluate the impact of fake feedbacks and the effect of the gossip protocol on the RMS under analysis, it is useful to consider the reputation hypothetically estimated by an ideal omniscient RMS that knows the true outcome of each transaction, and that can not be influenced by fake feedbacks.

Such an ideal RMS is implemented through a *truth-holder* agent (see Fig. 2.7), external to the agent network, that, at each simulation step, is responsible for collecting the truthful outcomes of agent interactions in order to build the ground-truth reputation $R_i^*(t)$ for each agent v_i. It is worth noting that the *truth-holder* is totally transparent to the RMS, and it only aims to provide a centralized tool to compute the error associated with the reputation estimated by the RMS.

In order to distinguish the bias introduced by fake feedbacks, it is convenient that the reputation aggregation algorithm adopted by the *truth-holder* is the same used by the network agents; thus, it must be redefined by the designers in order to meet the behavior of the RMS under analysis.

Thus, error on the single agent can be obtained as the difference between the *ground truth* and the *average reputation* of the agent v_i:

$$e_i^*(t) = |R_i^*(t) - r_i(t)|. \tag{2.17}$$

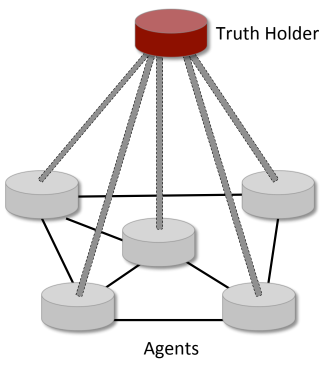


Figure 2.7: The role of the *truth-holder* process.

The mean value of these errors computed over the whole network represents the *average system error* due to fake feedbacks and to the gossip protocol:

$$e^*(t) = \frac{\sum\limits_{v_i \in V(t)} e_i^*(t)}{\#\left\{V(t)\right\}}. \tag{2.18}$$

This value can be used to evaluate the performance of the RMS since the first simulation steps.

2.4.2 Vulnerability Metrics

In order to measure the robustness of a RMS against the security attacks, DRESS allows developers to exploit a set of predefined metrics, or to define new ones according to the simulation needs.

By default, DRESS implements the following set of vulnerability metrics, preliminarily outlined in (Agate et al., 2016b):

- *time-to-falsify (TF)*: the time required to succeed in an attack aimed to falsify the reputation of a target agent;
- *exploitation-time (ET)*: how long a malicious agent can operate before its behavior is detected;
- *collusion-degree (CD)*: the number of agents required for a cooperative attack.

Each metric has a numeric score S, $0 \leq S \leq 10$, that corresponds to a qualitative rating according to the scale proposed by the Common Vulnerability Scoring System (CVSS) (CVSS, 2015): *none* (if $S < 0.1$), *low* ($0.1 \leq S < 4.0$), *medium* ($4.0 \leq S < 7.0$), *high* ($7.0 \leq S < 9.0$), *critical* ($9.0 \leq S \leq 10.0$).

In order to provide sound definitions of the vulnerability metrics, different *success conditions* are defined for each attack. Such conditions are valid for RMSs based on absolute reputation. Typically, these definitions are related to the long-term reputation of an agent, that is defined as the stable reputation estimated after a time threshold, T_{max}, typically corresponding to the simulation time and that can be set by the user. For instance, the promoting attack succeeds if the long-term reputation of a selfish agent is greater than half of the maximum reputation value R_{max}, i.e., $R_{th} = R_{max}/2$, while the slandering attack succeeds if the long-term reputation of a cooperative agent is below such limit.

The *collusion-degree*, *CD*, can be expressed as the percentage of malicious agents needed for completing an attack within a time that for instance can be set as $T_{th} = T_{max}/2$. For collaborative attacks, such as slandering and promoting, the vulnerability of the RMS can be measured by observing both the *time-to-falsify* *TF* and the *CD* values.

When dealing with attacks whose aim is the abuse of system resources for a period as long as possible, the most significant vulnerability measure is the *exploitation-time* ET. For the whitewashing attack, if we assume that T^* time steps are required for the reputation to go below the R_{th} threshold, the exploitation-time can be defined as $ET(whitewashing) = T^*/T_{max}$. Also the R_{th} threshold can be overwritten in order to define stricter or weaker success condition for attacks, and it aims only to define a common reference for evaluating different RMSs. Traitor attacks are performed by switching cooperative and selfish behavior for a certain number of time steps, T_{coop} and $T_{selfish}$ respectively. In such case, the exploitation-time is the lowest ratio $ET(traitor) = T_{coop}/(T_{coop} + T_{selfish})$ which guarantees that reputation remains above the R_{th} threshold.

The vulnerabilities of a RMS exposed to slandering (S), promoting (SP), whitewashing (W), and traitor (T) attacks can be expressed as:

$$\begin{aligned} v_S &= [1 - \mathit{TF}(slandering)] \times [1 - \mathit{CD}(slandering)]; \\ v_{SP} &= [1 - \mathit{TF}(self\text{-}prom)] \times [1 - \mathit{CD}(self\text{-}prom)]; \\ v_W &= ET(whitewashing); \\ v_T &= ET(traitor). \end{aligned}$$

The overall vulnerability index is a vector containing the list of the detected vulnerabilities, and the number of vulnerabilities evaluated as *high* or *critical*, i.e., $\#_{hc}$:

$$\mathit{RMS\_vulnerability} = \{[v^*_{SP}, v^*_S, v^*_W, v^*_T], \#_{hc}\},$$

where the v^* values are obtained by applying a gamma correction, with $\gamma = 0.5$, and a scale factor $K = 10$:

$$v^* = K \times v^\gamma.$$

K and γ values have been experimentally determined so that the four vulnerabilities metrics would be comparable in a network composed of 100 agents with 6 neighbors for each agent on average. DRESS users can modify these parameters in order to obtain a different scales for metrics.

Chapter 3

A General Purpose RMS

In order to better illustrate DRESS functionalities and to show the high degree of configurability that can be achieved, this chapter presents all the actions that a researcher must take, step by step, in order to implement their application scenario and their RMS. For this reason, first of all, an RMS possessing all the characteristics of a distributed reputation system is proposed. Moreover, it is described the way to configure the particular RMS to be analyzed, together with an explanation of how to execute the simulation using the functions provided by the simulation tool. Then, an analysis of reputation trend of the RMS agents, an analysis of accuracy of the proposed RMS and a comparison among different RMS policies are presented. Finally, a security analysis and an evaluation of the system scalability are shown.

3.1 The Proposed RMS

The RMS considered here adopts an absolute reputation model with $r_{ij}(t) \in [0, 1]$.

The number of available services and their distribution among providers can be specified through the simulation parameters. All the agents provide the same single service, and according to the implemented *service announcement protocol*, each agent v_i sends the list of provided services (in this case a singleton) only to its direct neighbors N_i.

Consumer agents use the *service selection* method to select a service among those available. In order to make the RMS analysis independent from a specific decision making mechanism, a simple service selection method which does not perform any specific choice was considerd, but rather it sends a service request to all the neighbors. Consequently, the whole neighborhood is uniformly explored, avoiding the bias potentially introduced by a specific policy, e.g., select the provider with the highest reputation.

The *service reply* method implements an *incentive mechanism* proposed in (Crapanzano et al., 2010), that randomly provides a reply with a probability which is proportional to the reputation of the requiring agent. The *behavior as service provider* further enforces such probability by a factor corresponding to the agent's cooperativeness degree, which is expressed as a value $\tilde{R}_i \in [0, 1]$, where a fully cooperative agent is characterized by $\tilde{R}_i = 1$, while a totally selfish agent by $\tilde{R}_i = 0$.

According to such method the probability $p_{ij}(t)$ that agent v_i provides a positive reply to service request coming from agent v_j at time t can be expressed as follows:

$$p_{ij}(t) = r_{ij}(t) * \tilde{R}_i(t). \tag{3.1}$$

After each interaction, the consumer agent stores in its transaction record a feedback $f_{ij}(t) \in \{1, 0\}$ depending on whether the service has been provided or not.

The components of the considered RMS are fully described by defining *local trust evaluation*, *gossip protocol* and *information fusion* methods.

The *local trust evaluation*, inspired to (Kamvar et al., 2003), computes the *local trust* of a given provider as the number of satisfactory transactions, over the total number of requests in the last ΔT time steps, as specified by the following equation:

$$l_{ij}(t) = \Lambda\left([f_{ij}(t')]_{t'=0:t}\right) = \frac{\sum\limits_{t'=t-\Delta T}^{t} f_{ij}(t')}{\#\left\{[f_{ij}(t')]_{t'=(t-\Delta T):t}\right\}}. \tag{3.2}$$

The *local trust* of the proposed RMS is based on feedbacks about the transactions in the previous 30 simulation steps, i.e., $\Delta T = 30$.

The next step consists in sharing the reputation values with other agents through the *gossip protocol*; in this sample case, information is exchanged with neighbor agents only, i.e., $v_k \in N_i(t)$, and protocol messages contain only last reputation values, as specified in the following equation:

$$\begin{aligned} \mathcal{M}_{i\to k}(t) &= \Gamma\left([f_{ij}(t')]_{\substack{t'=0:t-1 \\ v_j \in O_i(t-1)}}, [\mathcal{M}_i(t')]_{t'=0:t-1} \right) = \\ &= \Gamma\left([r_{ij}(t-1)]_{v_j \in O_i(t-1)} \right) = [(j, r_{ij}(t-1))]_{v_j \in O_i(t-1), v_j \neq v_k} . \end{aligned} \tag{3.3}$$

The *information fusion* method allows agents to merge their direct experience with information obtained through the gossip protocol. Such a method, proposed in (De Paola and Tamburo, 2008), states that gossiped information is weighted with reputation of the gossiper agents. In the sample case implementation, a parameter α specifies the weight of last direct experience with respect to past history, and a parameter β specifies the weight of received opinions on the overall reputation, as specified by the following equation:

$$\begin{aligned} r_{ij}(t) &= \Phi\left(r_{ij}(t-1), l_{ij}(t), \mathcal{M}_i(t)\right) = \\ &= (1-\beta) * [\alpha * l_{ij}(t) + (1-\alpha) * r_{ij}(t-1)] + \beta * \frac{\sum\limits_{v_k \in N_i(t)} r_{ik}(t-1) * r_{kj}(t-1)}{\sum\limits_{v_k \in N_i(t)} r_{ik}(t-1)}, \end{aligned} \tag{3.4}$$

where $\alpha, \beta \in [0, 1]$.

It is worth noticing that α and β, together with the initial default reputation value r_0, are declared as *varying parameters* so that they can be automatically tuned by the simulator in different simulation runs.

Once the RMS si fully defined, DRESS can be set to simulate different topologies and different security attacks, in order to obtain a quantitative evaluation of the system robustness.

In order to run a simulation, it is necessary to define the network topology G, and the behavior pattern for each node, through a set of configuration files.

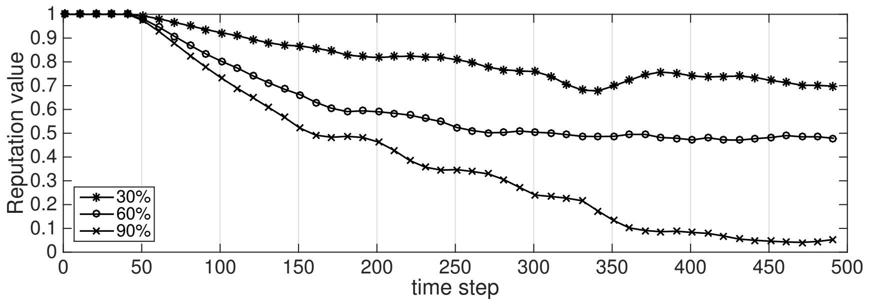


Figure 3.1: Simulation of slandering attacks. Reputation values of the victim agent as observed by neutral agents while varying the percentage of attacking agents.

3.2 Experimental Evaluation

Experimental results have been obtained by simulating a network composed of 100 nodes, where each node has 6 neighbors on average, randomly selected ensuring that the network is connected. The length of the simulation, i.e., T_{max}, has been set to 500 time steps.

3.2.1 Analysis of reputation trend

The first test focuses on using DRESS to analyze the reputation of a target of a slandering attack. To this end it is possible to use the proper DRESS interfaces to set the duration of the simulation, the target ID, and which information to observe, i.e., the reputation value. This preliminary test is characterized by the following parameters: the α factor, which weights the local trust to produce the local reputation, is $\alpha = 0.1$; the β factor, which weights the gossiped information to produce the final reputation, is $\beta = 0.1$. A relevant aspect to evaluate in this type of attack is how the number of attacking agents affects the reputation of the victim. We can intuitively state that the greater is the number of attacking agents, the more effective is the attack. However, DRESS may support this assumption by quantifying how many agents should be involved to finalize the attack. Fig. 3.1 is based on the results obtained from different simulation runs while varying the percentage of malicious agents in the network. It shows that as long as the 60% of the agents are hostile, the reputation of the victim decreases only by a factor of

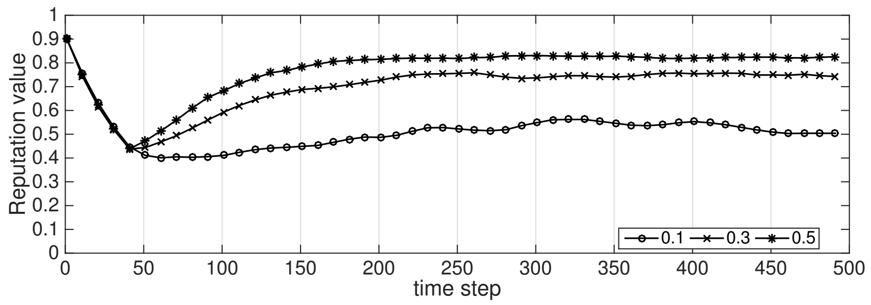


Figure 3.2: Simulation of promoting attacks. Reputation values of the victim agent as observed by neutral agents while varying the weight β of gossiped information.

0.5, whilst in order to tear down the reputation of the victim at least the 90% of the agents should exhibit a malicious behavior.

The same experimental setup can be easily adapted to simulate the promoting attack. In this case, the goal is to measure the behavior of reputation values of the selfish agent as observed by a trustworthy agent, while varying the weight of gossiped information. DRESS is able to automatically perform such an evaluation since the weight β has been defined as a *varying parameter* during the RMS implementation. Fig. 3.2 shows that a low weight assigned to gossiped information (i.e., 0.1) allows the observer to detect the selfish behavior mainly on the basis of direct interactions, whilst a higher weight (i.e., 0.5) makes that fake information, advertised through the gossip protocol, increases the reputation of the target agent.

Two more simulations have been performed in order to implement whitewashing and traitor attacks.

During a whitewashing attack, a malicious agent creates new accounts so as to obtain a default reputation value obfuscating its past selfish behavior. Evaluating the effectiveness of such an attack corresponds to analyze how the reputation of a new agent varies according to its behavior. To simulate this attack, it is necessary to specify in DRESS the time when a new agent should join the network. For example, Fig. 3.3 shows how the reputation of a new selfish agent that joins the network after 70 time steps changes varying the initial default reputation value. Different default reputation values assigned by the RMS to new agents impact on how long a malicious agent is able to abuse the community resources.

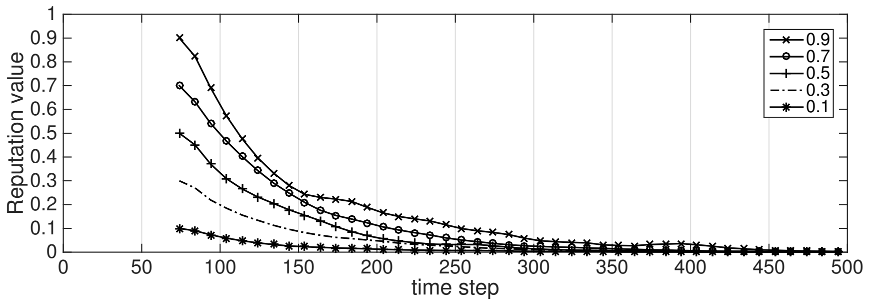


Figure 3.3: Simulation of Whitewashing attacks. Reputation values of a selfish agent that joins the network after 70 time steps, varying the initial reputation value assigned by the RMS.

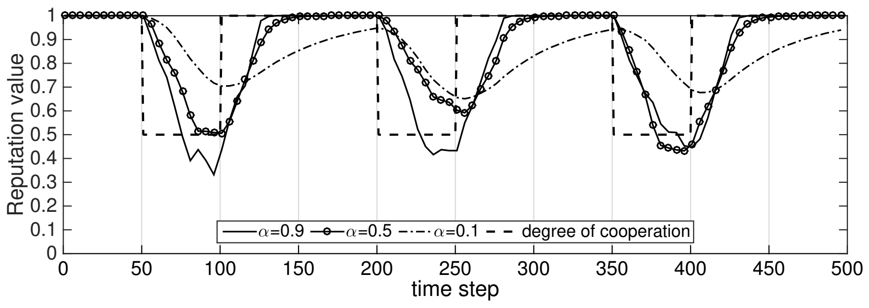


Figure 3.4: Simulation of traitor attacks. Reputation values of a traitor agent which alternates cooperative and selfish behaviors, varying the α factor that weights the recent experience to build the local reputation.

Simulation of a traitor attack requires to define a complex behavior where a given agent alternates cooperative and partially cooperative behavior, i.e., by satisfying all or only part of the received requests respectively. DRESS allows the easy composition of such a behavior and also the description of duration of each atomic behavior. In the experimental setting adopted here, traitor agents alternate fully cooperative and partially cooperative behavior, i.e., $\tilde{R}_i(t) = 0.5$, for regular time intervals, i.e., $\Delta T_1 = 50, \Delta T_2 = 100$ time steps. Exploiting such configurations, a test is performed to evaluate how different values of the weight for recent experience, α, speed up the detection of variations of the agent behavior. Fig. 3.4 shows the results of simulating a traitor attack when a malicious agent alternates a fully cooperative and a partially cooperative behavior. As expected, RMSs with higher values of α detect earlier a change in agent behavior.

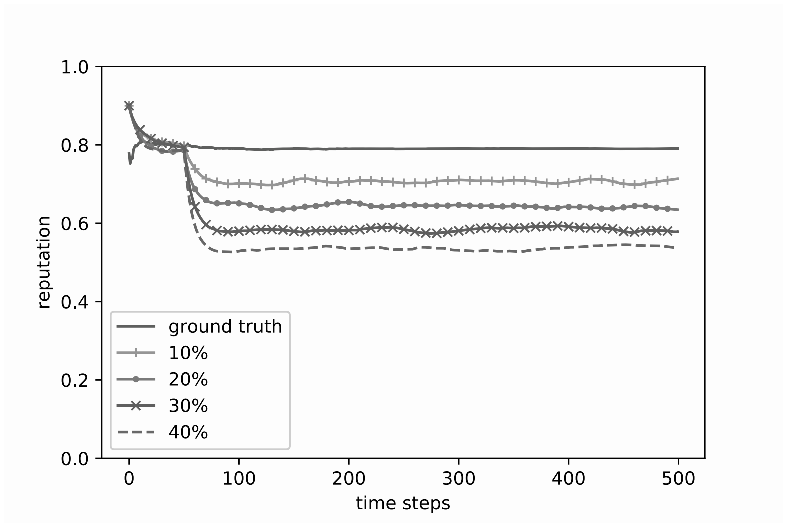


Figure 3.5: Comparison between the reputation of the victim agent v_i computed by the *truth-holder*, i.e., $R_i^*(t)$, and the average reputation estimated by the distributed RMS, i.e., $r_i(t)$, while varying the percentage of malicious agents involved in the slandering attack.

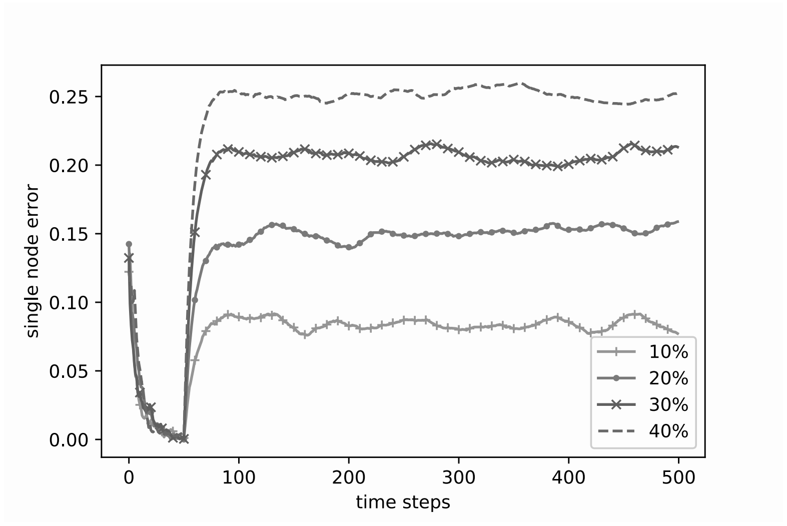


Figure 3.6: *Average system error*, i.e., $e_i^*(t)$, of the RMS considered as sample case during a slandering attack against the victim agent v_i, while varying the percentage of malicious agent involved.

3.2.2 Analysis of accuracy

Through the accuracy metrics adopted by DRESS it is possible to quantify the error performed by the RMS under evaluation. In particular, the average system error quantifies the deviation from the *truth-holder*'s estimation, as defined in Section 2.4.1. In this sample case the *truth-holder* computes the ground-truth reputation of the agent v_i, i.e., $R_i^*(t)$, according to the local trust evaluation algorithm:

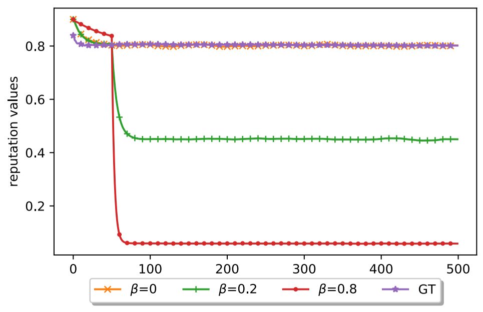
$$R_i^*(t) = \frac{\sum\limits_{v_j \in V(t)} \sum\limits_{t'=t-\Delta T}^{t} f_{ij}(t')}{\# \left\{ [f_{ij}(t')]_{\substack{t'=(t-\Delta T):t \\ v_j \in V(t)}} \right\}}. \tag{3.5}$$

To show the analysis supported by DRESS through the available metrics, an experiment was performed in a network where, for each agent, the *behavior as service provider* is obtained by setting a *cooperative degree* equals to 0.8, and where a slandering attack is performed after 50 time steps (in a simulation of 500 time steps).

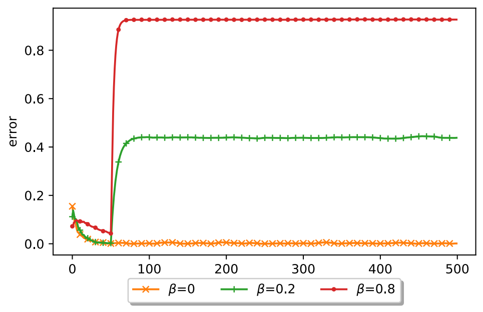
Fig. 3.5 compares the ground-truth reputation and the average reputation estimated by the distributed RMS, while varying the percentage of malicious agents. As we can observe, the reputation estimated by the RMS deviates from ground truth as much as the percentage of malicious agents increases. Specularly, Fig. 3.6 shows the average system error of the considered RMS, evaluated as the average difference between the reputation value estimated by the RMS and the corresponding ground truth. Results suggest that the estimation error made by this specific RMS under a slandering attack grows quite proportionally to the percentage of malicious agents. Thus, it does not exhibit an *amplification* of the fake negative opinions, thanks to the inclusion of the direct experience (i.e., local trust) in the reputation aggregation algorithm.

3.2.3 Comparisons among different RMS policies

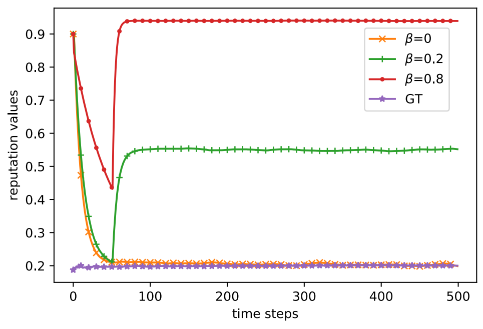
In the following set of experiments it is shown how the proposed platform allows to compare different RMS policies by evaluating their accuracy and resistance to forged feedbacks, given the same network configuration. The reputation network for this set consists of 300 agents, where 30% of them are implicated to perform a slandering attack against a 10% of cooperative agents. This experiment compares



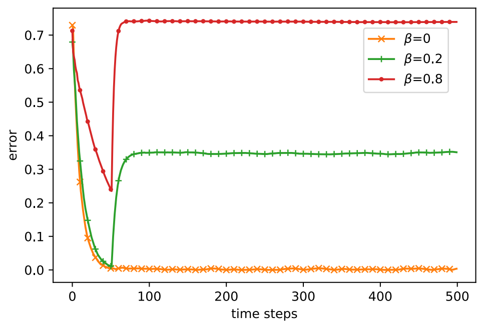
(a)



(b)



(c)



(d)

Figure 3.7: Reputation (a, c) and average system error (b, d) measured while simulating slandering (a, b) and promoting (c, d) attacks performed on a network of 300 nodes with 30% of implicated agents. The different curves are obtained while varying the parameter β which weights gossip information with respect to the local trust. The GT curve shows the reputation computed by the *truth-holder*.

three policies obtained by varying the parameter β of the RMS considered as case study. The first policy uses only direct experience to estimate agents' reputation, i.e., $\beta = 0$. The second policy is characterized by a good balance between local trust and gossip information, and is obtained by setting $\beta = 0.2$. The third policy relies excessively on gossip information, with a weight $\beta = 0.8$. Fig. 3.7a compares the reputation computed by the *truth-holder* and the average reputation estimated by these policies, while Fig. 3.7b shows the corresponding average system errors. As expected, smaller weights to gossip information reduce the vulnerabilities to attacks characterized by the injection of false information. In the borderline policy

	Parameter	Description	Value
Topology	N	Number of agents	100
Varying Parameters	α	weight of recent experience	0.1
	β	weight of gossiped information	0.1
	rep_0	default initial reputation value	0.9

Table 3.1: Simulation parameters of the RMS considered as sample case.

Metric	Value	Index	Value
$TF(slandering)$	0.71	v_S^*	1.78
$CD(slandering)$	0.65		
$TF(self\text{-}prom)$	0.79	v_{SP}^*	3.18
$CD(self\text{-}prom)$	0.85		
$ET(whitewashing)$	0.054	v_W^*	2.32
$ET(traitor)$	0.25	v_T^*	5.00

Table 3.2: Security evaluation of the RMS considered as sample case.

where gossiped information are not considered at all, the estimated reputation corresponds to the ground truth, thus obtaining an average error which quickly goes to zero.

The same analysis can be performed considering promoting attacks, as shown in Fig. 3.7c and 3.7d. In this case, using the same network configuration described above, promoting attacks are performed to raise the reputation of a group of target agents whose real cooperativeness is 0.2. As shown before, policies that limit the weight of gossiped information are characterized by greater resistance.

3.2.4 Analysis of security

Besides accuracy analysis, DRESS allows to perform a global evaluation of the considered RMS, according to the chosen set of security metrics. As described in section 2.4.2, the base set of metrics of DRESS includes *time-to-falsify (TF)*, *exploitation-time (ET)*, and *collusion-degree (CD)*, which are summarized in a

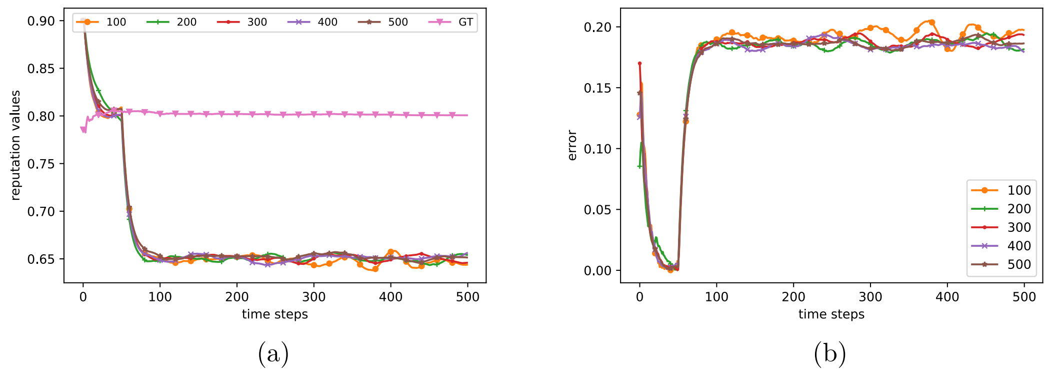


Figure 3.8: Reputation (a) and average system error (b) measured while simulating a slandering attack in which the 20% of the network is implicated and all the agents have a cooperativeness degree of 0.8. The different curves are obtained in networks composed of 100, 200, 300, 400, and 500 nodes. The *GT* curve shows the reputation computed by the *truth-holder*.

global vulnerability index. Researchers can analyze such index in order to evaluate the impact of the RMS design choices on its vulnerability to security attacks.

For the RMS considered as sample case, with the parameters specified in Table 3.1, DRESS produces the set of indexes shown in Table 3.2, and the following global vulnerability index:

$$\begin{aligned} RMS\_vulnerability &= \{v^*_{SP}, v^*_S, v^*_W, v^*_T], \#_{hc}\} \\ &= \{[1.78, 3.18, 2.32, 5.00], 0\}\,. \end{aligned}$$

These results show that the RMS does not exhibit any high or critical vulnerability. In particular, the design choices produce a greater robustness to promoting attacks than to slandering attacks, a low vulnerability to whitewashing attacks, and a medium vulnerability to traitor attacks.

3.2.5 System scalability

Some tests were run to verify the capability of the proposed platform of measuring the actual performance of a given RMS, regardless of the size of the reputation network. In particular, a slandering attack launched by a set of implicated agents

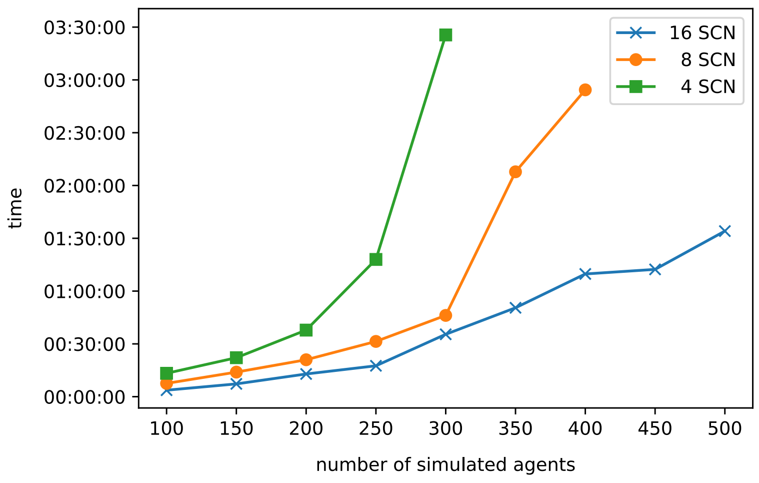


Figure 3.9: Simulation time using 4, 8, and 16 single-core nodes.

(the 20% of the network) against other agents (the 10% of the network) that have a cooperation degree of 0.8 was considered. Fig. 3.8a compares the reputation computed by the *truth-holder* and the average reputation estimated by the RMS on networks of 100, 200, 300, 400, and 500 nodes. Fig. 3.8b shows that the corresponding average system errors are quite similar, so proving that the diagnostic capability of the simulator is not dependent on the size of the network.

Other tests have been performed to measure how the simulation time depends on the number of deployed computational nodes. In particular, three experiments were run considering clusters of 4, 8, and 16 single-core nodes (SCNs). Results from Fig 3.9 show that the number of SCNs deeply impacts on the number of agents the platform can simulate. In particular, the current implementation of the simulator is not able to support more than 300 agents when using 4 SCNs, while in order to simulate a network of 500 agents at least 16 SCNs are needed. This is mainly due to the inter-process communication routines needed to support the simulation, according to which every agent exchanges message with all other agents in the network, regardless of the network topology. Nevertheless, for any number of SCN, the simulation time exhibits a quadratic growth, which is quite reasonable to simulate reputation networks of significant dimensions.

Chapter 4

A Domain Specific RMS

Reputation management systems can make a strong contribution to the recognition of reliability in many application contexts, such as the sharing of electricity from renewable sources. Distributed and pervasive renewable energies, especially photovoltaics (PV), are expected to be the future of power systems, as envisioned by the paradigm of Virtual Power Plants (VPPs) recently proposed by several research and governmental bodies (Johnson et al.; Asmus, 2010; Vasirani et al., 2013; Hernández et al., 2013; Palizban et al., 2014). The goal of VPPs is to enable renewable smart grid systems which are capable of delivering reliable ancillary services, traditionally provided by large power plants.

VPP will enable a two-way flow of electricity and information (Wang et al., 2017). Hence, users and providers will have the opportunity to build up a constructive interaction unthinkable in a standard power system. Thanks to this interaction, and the diffusion of renewable energy sources at the consumer level, users become the so-called *prosumers* (Parag and Sovacool, 2016), as they would be able not only to consume, but also to produce energy at their premises. These new advancements pave the way to the advent of *energy sharing systems* (ESS), where users are able to exchange energy between themselves to reduce their energy bill (Liu et al., 2017). In this scenario, having a precise idea of the degree of collaboration of users and their ability to respond positively to energy transactions seems to be of the utmost importance. DRESS was used to simulate both an electricity sharing system and a reputation calculation system for the identification of the most cooperative users, proving to be an extremely flexible environment for the design of a new RMS.

Therefore, an extensive introduction to the problem is presented, together with a study of existing approaches for energy sharing systems, and a brief summary of the main contribution made. Moreover, the system model and problem statement are described. Finally, in order to address the energy sharing optimization problem, a possible RMS-based solution is proposed, and the relative experimental results are discussed.

4.1 The Energy Sharing Application Domain

To motivate and support the potential of residential customer participating in an ESS, Fig. 4.1 shows a typical daily household energy consumption and production with a 8kW PV panel[1]. Clearly, there is a mismatch between when the energy is produced and when it is consumed. Currently in the U.S., the excess energy is either wasted or sold to the utility company for low prices, depending on the state regulations[2,3]. Batteries with sufficient capacities would solve this inefficiency, but they are not yet commonly adopted due to their high costs. In fact, it has been shown that each home should be equipped with batteries larger than 12kWh to store sufficient energy, costing more than $\$6,000$ per household (Zhu et al., 2011). As a result, supported by the emerging paradigm of VPP, a viable and more attractive alternative is to exchange excess energy between users through an ESS. An additional benefit of ESS is the reduced loss incurred in energy transfer resulting from the closer proximity of users' homes with respect to the utility company. An example of a commercial application of ESS is the Dutch start-up Vandebron. Vandebron enables the local renewable electricity generators to sell their energy under an online peer-to-peer marketplace platform independent of any utility or government agency[4].

[1]Pecan street inc. dataport 2014. URL www.pecanstreet.org.

[2]Serc: State environmental resource center, 2011. URL http://www.serconline.org/netmetering/stateactivity.htm.

[3]Freeing the grid: Best and worst practices in state netmetering policies and interconnection procedure, 2009. URL http://www.newenergychoices.org/ uploads/FreeingTheGrid2009.pd.

[4]Vandebron. URL https://vandebron.nl/.

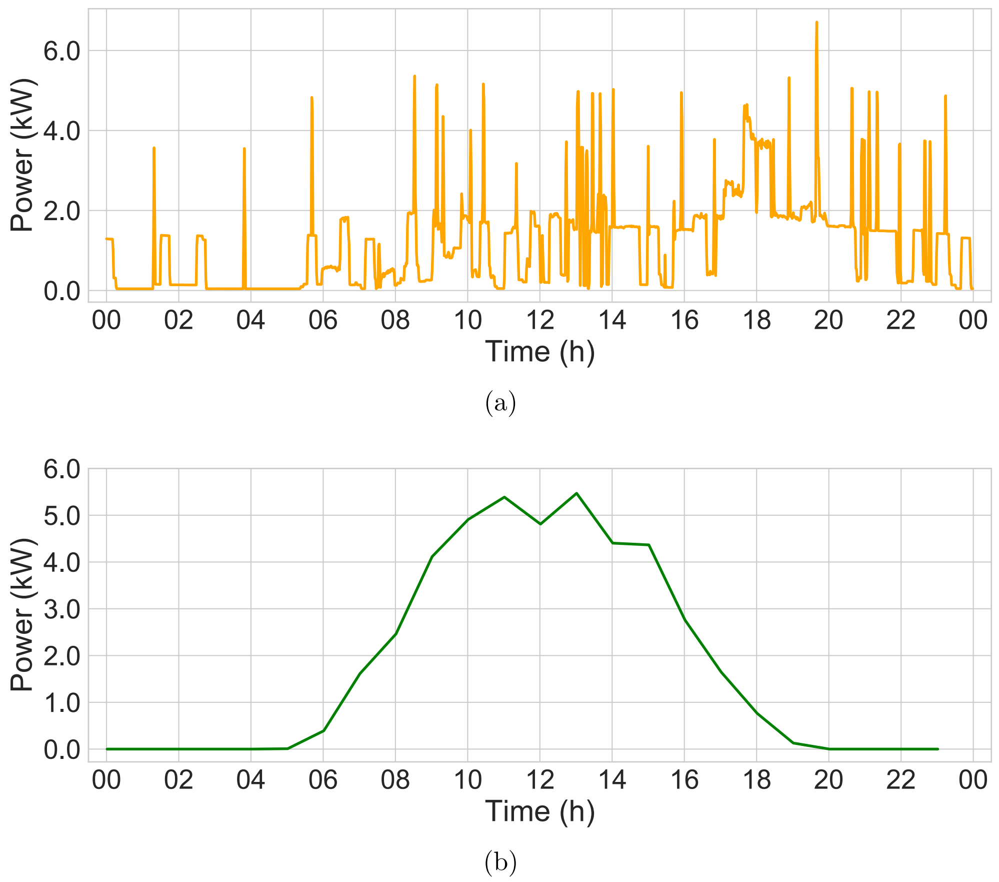


Figure 4.1: Mismatch between energy consumption (a) and PV generation (b).

Previous research in ESS, such as (Liu et al., 2017; Zhu et al., 2013; Althaher et al., 2015; Jhala et al., 2018), is based on simplified models of human behavior, for example assuming that users are always available and engaged with the ESS, or will always follow the suggestions that the ESS would recommend. Recent research in the social science domain has shown that, in fact, users have significant heterogeneity in their preferences for energy sources (Contu et al., 2016) as well as in engaging with energy management systems in general (Ropuszyńska-Surma and Węglarz, 2018). As a result, previous work in ESS may fail when implemented in the real world (Dolve et al., 2018; Silvestri et al., 2017).

This part of the thesis studies a peer-to-peer network topology in which each agent, as a producer or consumer, is able to trade energy with its neighbors. To accommodate an interactive energy sharing as a multiuser system, different

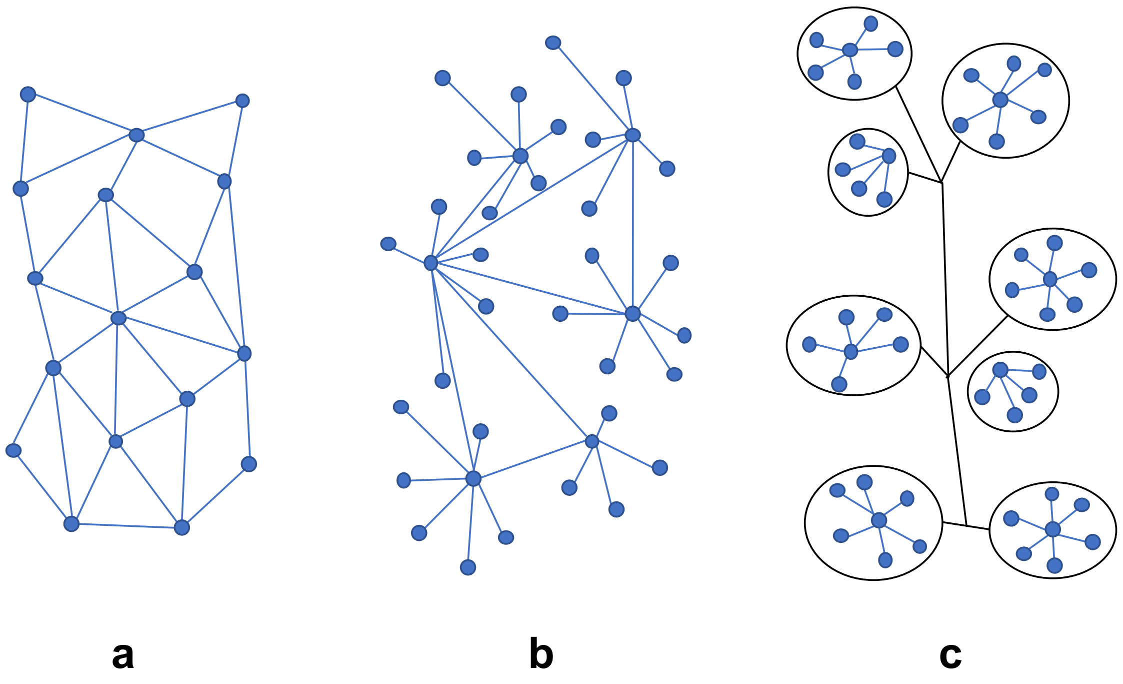


Figure 4.2: Network topologies: (a) peer-to-peer model, (b) prosumers connected to micrograms and (c) prosumers group model (Parag and Sovacool, 2016).

network structures can be assumed. For instance, (Parag and Sovacool, 2016) investigates three network models in which the prosumers are connected (i) directly to each other, (ii) to a microgrid, or (iii) form community groups. A schematic of such topologies is depicted in Fig. 4.2.

The work in this thesis advances the state-of-the-art in ESS by considering realistic and heterogeneous user behaviors in terms of preferences and engagement. Specifically, here an energy community which implements an ESS is considered. Within this platform, users are allowed to sell and buy energy to and from other members in the community, as well as from renewable and standard power plants. This scenario can be seen as a community in a microgrid connected to a larger smart grid, and it is grounded on previous models proposed for ESS (Parag and Sovacool, 2016). An active user participation in the energy exchange is envisioned, where users may have different preferences for different energy sources (e.g., renewable, nuclear, coal, etc.), as well as a different level of engagement with the system.

According to the proposed approach, periodically (e.g., daily) the ESS calculates for each user a prediction of the amount of needed energy, and based on such prediction it matches production and consumption to maximize the system performance given the users preferences and level of engagement. The matching is translated into a *personalized recommendation*, sent for example through a smartphone app. This recommendation includes a list of energy sources and the amount that should be bought from each source to fulfill the needs. If a recommendation is accepted by the user, it needs to be honored by the system. Conversely, if a user ignores a recommendation, for example because he/she is not engaged with the ESS, or because the source of energy does not match his/her preferences, the committed energy is wasted due to the limited energy storage at the producer side. As a result, in order to maximize the system performance, it is fundamental to take into account the user behavior while matching the produced and consumed energy. In this dissertation this matching problem is formulated using Mixed Integer Linear Programming (MILP). The problem aims at maximizing the amount of exchanged energy, while taking into account the user preference as well as physical constraints imposed by the loss of energy in the transfer process. The considered problem is NP-Hard, and furthermore it requires prior knowledge of the user behavioral model. For this reason, it was proposed an heuristic based on reinforcement learning that trades off exploration and exploitation in order to learn the user preference and used as reputation values, while optimizing the system performance (Gai et al., 2012). The proposed algorithm has been shown to have a *bounded regret* with respect to the optimal case in which the user preference is known.

The proposed system was compared with an existing approach for ESS proposed in (Zhu et al., 2013), using simulations based on real traces for consumption and production of energy. Results show that the proposed approach is able to effectively learn the user preference and significantly improve the performance of the system.

In summary, the main contributions are the following:

- The problem of optimizing the performance of an ESS is addressed while considering realistic user behavioral models in terms of preferences and engagement.

- The problem is formulated using MILP and is shown to be NP-Hard.
- An heuristic based on reinforcement learning is proposed to learn the user behavioral model while optimizing the system performance.
- The proposed approach is compared with state-of-the-art solutions using simulations based on real traces. Results show that the proposed solution significantly outperform existing approaches by effectively learning the user preferences.

The idea of Virtual Power Plants (VPPs) and distributed energy systems based on renewable energy resources has attracted significant interest from both the academic and industry community (Ackermann et al., 2001; Asmus, 2010; Vasirani et al., 2013; Hernández et al., 2013; Palizban et al., 2014; Rehmani et al., 2018). Authors in (Lakshminarayana et al., 2014) studied the tradeoff between the use of storage and the concept of distributed generation. The goal was to reduce the energy fluctuation due to the integration of renewable energy generation into smart grids. Their results showed that in absence of large storage, the grid can notably gain from exchange energy between the users. A DC power sharing among nearby homes has been introduced by the authors of (Zhu et al., 2013) to address the problem of mismatching between energy harvesting and consumption in microgrids. In a recent work, game theoretic and hierarchical optimization approaches have been used to minimize the power mismatch in and among microgrids in a multiagent-based energy market (Esfahani et al., 2019). Note that, none of the above mentioned papers considers aspects of user preference, thus assuming users to be either extraneous to the system or perfectly complaint with the system decision.

Modeling user behavior in smart grids has been considered in the context of *Demand response* (DR) (Commission, December 17, 2015). DR has been a major research effort mainly focused on preventing the occurrence of demand peaks, where the price of electricity is changed dynamically to alter the user behavior. The authors of (Jhala et al., 2018) use a reverse approach, in which prospect theory is used to model the user response to energy prices, and focus on the impact of such realistic behaviors on the system. Although relatively easy to implement, thanks to the diffusion of advanced metering infrastructure (AMI) (Bhattacharjee et al., 2017), DR adoption rates are low (Sioshansi, 2019 (accessed February 3, 2020), and its effectiveness is not clear as it can even lead to an increase of energy consumption (Earle and Faruqui, 2006; Herkert and Kostyk, 2015; Delmas et al., 2013).

Differently from previous work, this thesis focuses on learning and integrating realistic user preferences and related reputations in the optimization of ESS. To the best of found knowledge, this is the first effort that combines optimization, machine learning and user behavioral modeling in the context of ESS.

4.2 System Model and Assumptions

Let us consider a set of users U. This set includes users equipped with on-site power generators such as PV panels, users without power generation capabilities, as well as larger power plants based on renewable energies (e.g., solar, wind, etc.) and standard power plants (coal, nuclear, hydroelectric, etc.). Each producer may sell energy at a different cost, which is assumed does not change over time. For convenience, let us partition the set U in a set of m producers $B = \{b_1, b_2, \ldots, b_m\}$ and a set of n consumers $C = \{c_1, c_2, \ldots, c_n\}$. Note that, consumers may include residential users with power generation capabilities unable to fulfill their daily energy needs. Let us envision that energy exchanges are performed daily, for example during the evening for the next day. The ESS estimates for each producer b_i the next-day production capacity of R_i, with $i = 1, 2, \ldots, m$, and for each consumer c_j the energy requirement of w_j, for $j = 1, 2, \ldots, n$, expected for the next day. It has been shown that these can be accurately predicted with time-series analysis techniques, such as exponential moving average (Zhu et al., 2013).

Let us consider an ESS in which users, and specifically consumers, have an active role in the exchange process. Specifically, the ESS sends daily to each customer through a smartphone app a *personalized recommendation.* This recommendation consists of a list of producers, the amount of energy to be bought from each of them, and the cost. Differently from previous works in this area, which assume users to be always compliant and engaged with the system, let us consider a realistic user behavioral model in which users may accept, reject or ignore each producer listed in the recommendation. This behavior is dictated by the consumer level of engagement with the ESS and by their preferences for the source of produced energy (e.g., coal, renewable, nuclear, etc.) and the price at which energy is sold by a producer. Let us model this preference through a probability $p_{ij} \in [0, 1]$ representing the likelihood that consumer c_j would like to buy energy from producer b_i. This probability is initially unknown, and to learn it a reinforcement learning approach is proposed.

This same probability represents a very interesting information for the choice of the most willing users to accept the transactions and from a different point of view it indicates the reputation that a certain user has inside the sharing system.

Let us consider that the recommendation list cannot exceed a maximum length K and the energy exchanged between two users should be greater than a minimum value α. These assumptions take into account the fact that it is unfeasible for users to manually accept a very long list of suggestions, and it is not convenient to exchange infinitesimal amounts of energy.

Note that, if a recommendation is accepted, the ESS will honor this exchange. As a result, a recommendation is a *commitment* of energy resources. Consequently, if a recommendation is rejected or ignored, it will result in an energy waste (or in energy sold to the utility company for a much lower price). As a result, the recommendations need to be carefully designed to maximize the performance of the system.

Finally, let us assume that when producer b_i sells energy to consumer c_j, there is an *energy loss* during the energy transfer (Zhu et al., 2013). This loss depends on the physical distance between b_i and c_j and it is directly proportional to the amount of energy exchanged. The loss is thus modeled as a fraction $L_{ij} \in [0, 1]$ of the energy exchanged. Let us assume that there is a maximum tolerable loss L_{max} that the ESS allows.

4.3 Problem Formulation

The goal of the ESS optimization problem is to find the recommendations to be sent to the customers so that the expected energy exchanged is maximized. This results in minimizing the amount of wasted energy. The problem is formulated in Eqs. (4.1)-(4.7).

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} w_j p_{ij} x_{ij} & & (4.1)\\
\text{s.t.} \quad & \sum_{j=1}^{n}(1+L_{ij})w_j x_{ij} \leq R_i, & \forall i \quad & (4.2)\\
& \sum_{i=1}^{m} x_{ij} \leq 1, & \forall j \quad & (4.3)\\
& \sum_{i=1}^{m} z_{ij} \leq K, & \forall j \quad & (4.4)\\
& \alpha z_{ij} \leq w_j x_{ij} \leq w_j z_{ij}, & \forall i,j \quad & (4.5)\\
& z_{ij} \geq x_{ij}, & \forall i,j \quad & (4.6)\\
& x_{ij} \in [0,1],\ z_{ij} \in \{0,1\}, & \forall i,j \quad & (4.7)
\end{aligned}
$$

The decision variables of the problem are $x_{i,j} \in [0,1]$. Given the energy demand w_j of consumer c_j, $x_{i,j}$ is the fraction of w_j that c_j is being recommended to buy from producer b_i. The goal is to maximizes the expected amount of exchanged energy, considering the probability p_{ij} that c_j will accept the recommendation. Let us also introduce a binary decision variable $z_{ij} \in \{0,1\}$, which is equal to 1 if $x_{i,j} > 0$, i.e. if b_i is included in the recommendation of c_j. The variables z_{ij} are used in the constraint in Eq. (4.4) to make sure that the recommendation has a length limited by K.

The constraint in Eq. (4.2) guarantees that the production capacity of producer b_i is not exceeded, considering also the loss that is incurred in the transmission. Similarly, constraint (4.3) ensures that the need of consumer c_j are not exceeded. Finally, Eq. (4.5) ensures that an exchange is larger than the minimum allowed amount α, and Eqs. (4.6)-(4.7) define the domain of the decision variables. Note that, the problem allows exchanges between all pairs of producers and consumers, given the problem constraints. Nevertheless, an additional constraint can be added to prevent losses above the maximum allowed fraction L_{max} by setting $x_{ij} = 0$ if $L_{ij} > L_{max}$.

Notation	Description
m	Number of producers
b_i	ith producer
R_i	Production capacity of ith producer
n	Number of consumers
a_j	jth consumer
w_j	Energy requirement of jth consumer
t	Time index
P_{ij}	Random variable corresponding to preference of consumer j buying from producer i
p_{ij}	Mean of P_{ij}
$\widehat{p}_{ij}$	Estimation of p_{ij}
m_{ij}	Number of suggesting producer j to consumer i
L_{ij}	Transmission loss rate between producer j and consumer i
$\mathbf{A}$	Action matrix

Table 4.1: Notation Summary

Table 4.1 summarizes the notation.

The following theorem shows that the problem is NP-Hard.

Theorem 1. *The optimization problem in Eq.* (4.1) *is NP-Hard.*

Proof. This proof provides a reduction from the Generalized Assignment Problem (GAP) (Fleischer et al., 2006). In a general instance of GAP, there are n tasks and m processors. Processor i has R_i resources. By assigning task j to processor i, a profit f_{ij} and consume g_{ij} resources can be obtained. A task can be assigned to a single process, and the goal is to find the assignment that provides the maximum profit given the resources of the processors.

The GAP is formulated as below.

$$\begin{aligned}
&\text{maximize} && \sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}x_{ij} && && (4.8)\\
&\text{s.t.} && \sum_{j=1}^{n} g_{ij}x_{ij} \le R_i, && \forall i && (4.9)\\
& && \sum_{i=1}^{m} x_{ij} = 1, && \forall j && (4.10)\\
& && x_{ij} \in \{0,1\}. && \forall i,j && (4.11)
\end{aligned}$$

From a general GAP instance, an instance of the problem can be created as follows. It is possible to consider a consumer for each task and a producer for each processor. K is set to $K = 1$, so that the recommendation for a consumer can contain at most a single producer. Furthermore, it is considered $(1 + L_{ij})w_j = g_{ij}$ and the energy production of producer i as R_i. Let us considerer $L_{max} = \infty$ so that all exchanges are possible.

At this point, the only difference between the reduced problem and the GAP problem is that the decision variables x_{ij} are continuous, while the decision variable under GAP are discrete. However, infinitesimal exchanges are not allowed in the proposed system, as they need to be greater than or equal to α. By setting $\alpha = w_j$, the constraint in Eq. (4.5) forces the decision variable x_{ij} to coincide with the discrete variable z_{ij}.

As a result, the solution of the reduced problem provides the assignment that maximizes the profit within the constrained processors' resources. Therefore, the considered problem is at least as hard as GAP, and thus it is NP-Hard. □

Note that, although the problem is NP-Hard the scale of the problem is limited by the constraint on the maximum loss. In fact, the complexity of the problem is proportional to the number of possible producer/consumer pairs. However, the loss in an energy transfer depends on the physical distance between producer and consumer, as a result the number of actual possible pairs is limited. In performed experiments, an optimizer such as Gurobi [5] is able to solve the problem in a very short time in all the considered settings.

[5]Gurobi Optimizer Reference Manual, 2015-2014.

However, the solution of such optimization problem requires the knowledge of the user preferences (p_{ij}), the expected production (R_i), and the expected consumption (w_j). As mentioned, the latter two can be predicted using time series analysis (Zhu et al., 2013). Conversely, learning the user behavior is challenging, as users may significantly differ in their preferences and engagement with the ESS (Contu et al., 2016; Ropuszyńska-Surma and Węglarz, 2018). For these reasons, a reinforcement learning based RMS to learn user preferences was developed. Since user preferences reveal valuable information about their ability to accept transactions, an RMS that uses preference informations as reputation informations was used.

4.4 A Reinforcement Learning Approach for User Preference Learning

A possible way of predicting the expected user preference is to directly ask users when the ESS is installed in their homes. However, social behavioral studies show that such information does not always reflect the actual preferences. Such situations typically occur when users make choices that are not always motivated by a well-defined logic, such as in the addressed case (Kahneman, 2003).

In order to learn the user reputation and engagement, i.e. the probabilities p_{ij} of the optimization problem, the user preference over time is observed and their reputation is built as consequence. This is a typical problem of exploration and exploitation in which the goal is to explore by sending recommendations to users to learn their preference, but at the same time optimize the system performance by exploiting what the algorithm has already learned, i.e. maximizing the exchanged energy by sending recommendations to users that are most likely going to accept. This problem is well modeled by reinforcement learning (Gai et al., 2012), as explained below, and this solution represents a useful and interesting Reputation Management System to establish those most willing to accept the energy transactions of users within the proposed energy sharing system.

Let us model the preference of consumer c_j, with respect to accepting a recommendation for buying energy from producer b_i, as a random variable P_{ij}. The realization of such variable at day t is referred to as $P_{ij}(t) \in \{0,1\}$. The mean value of P_{ij} is denoted as p_{ij}, that represents at the same time the reputation of j in accepting energy by i, and it is initially unknown. Let us also assume P_{ij} evolves as an i.i.d. process over time. Given the energy consumption/production predictions for day t, the ESS decides which suggestions should be sent to the consumers. This is modeled by an *action* matrix $\mathbf{A}(t) = [a_{ij}(t)]_{n\times m}$, where $a_{ij}(t) \in [0,1]$ represents the fraction of w_j that customer c_j is suggested to buy from b_i, similarly to the x_{ij} variables of the optimization problem. As a result, the total number of unknown variables is $N = n \times m$. The solution space $\mathcal{F}$ includes all feasible action matrices that would satisfy all the constraints of the optimization problem.

Similar to the optimization problem, maximizing the amount of exchanged energy is the goal. At each iteration of the optimization phase t, the ESS chooses the action matrix $\mathbf{A}(t)$ that maximizes the optimization function given the current knowledge. This knowledge is represented by the estimated averages $\widehat{p}_{ij}(t)$ for each random variable P_{ij}. For an action matrix $\mathbf{A}(t)$, the *reward* is defined as $\mathbf{R}_{\mathbf{A}(t)}(t) = \sum_{i,j} w_j a_{ij}(t) P_{ij}(t)$. Given the initially unknown distribution of the variables P_{ij}, the goal is to find a policy (i.e., series of action matrices in $\mathcal{F}$) that minimizes the *regret* up to the current time t, i.e. the difference between the expected reward having perfect knowledge of the variables realizations and that obtained by a given policy. Formally, the regret is expressed as $\mathcal{R}(t) = t\mathbf{R}^*_{\mathbf{A}(t)}(t) - \mathbb{E}[\sum_{t'=1}^{t} \mathbf{R}_{\mathbf{A}(t')}(t')]$, where $\mathbf{R}^*_{\mathbf{A}(t)}(t)$ is the reward obtained with perfect knowledge of the users' preferences.

Minimizing the regret is a hard problem, given the initial unknown variable distribution. However, an efficient algorithm based on reinforcement learning that ensures a *bounded* regret with respect to the optimal (Gai et al., 2012) can be formulated. The pseudo-code of the algorithm is shown Algorithm 4.1: User Preference Learning (UPL), and it is composed of two consecutive phases: *initialization* and *optimization*. During the initialization phase, N actions are played randomly in order to observe all the N random variables at least once.

In the optimization phase, the system plays an action that maximizes the function defined in line 8 of Algorithm 4.1, over the solution space $\mathcal{F}$. This can be accomplished by solving an optimization problem with constraint as in Eqs.

Algorithm 4.1 User Preference Learning (UPL)

```
1: //INITIALIZATION PHASE
2: for $r = 1, \dots, n$ and $q = 1, \dots, m$ do
3:     Select any $\mathbf{A} \in \mathcal{F}$ s.t. $\mathbf{A}_{rq} = 1$ ;
4:     Update $[\widehat{p}_{ij}]_{n\times m}$ and $[m_{ij}]_{n\times m}$;
5: //OPTIMIZATION PHASE
6: while True do
7:     $t = t + 1$
8:     Select an action $\mathbf{A}$ s.t.
       $\mathbf{A}(t) = \arg\max_{\mathbf{A}\in\mathcal{F}} \sum_{i=1}^{m}\sum_{j=1}^{n} w_j a_{ij}\left(\widehat{p}_{ij} + \sqrt{\frac{(N+1)ln\ t}{m_{ij}}}\right)$
9:     Update $[\widehat{p}_{ij}]_{n\times m}$ and $[m_{ij}]_{n\times m}$;
```

4.2-4.7, but with the following objective function:

$$\mathbf{A}(t) = \arg\max_{\mathbf{A}\in\mathcal{F}} \sum_{i=1}^{m}\sum_{j=1}^{n} w_j a_{ij}\left(\widehat{p}_{ij} + \sqrt{\frac{(N+1)ln\ t}{m_{ij}}}\right).$$

The optimization problem solved at time t is based on the estimation of the average values p_{ij} at time $t-1$, denoted as $\widehat{p}_{ij}(t-1)$. If the selected action at time t includes an energy transaction between consumer c_j and producer b_i, i.e., $a_{ij}(t) \neq 0$, the system observes a new realization $P_{ij}(t)$ of the random variable P_{ij}. The system uses this information to update the current knowledge estimation of $\widehat{p}_{ij}(t)$, as well as the total number $m_{ij}(t)$ of observations of the variable P_{ij}, as follows:

$$\widehat{p}_{ij}(t) = \begin{cases} \frac{\widehat{p}_{ij}(t-1)m_{ij}(t-1)+P_{ij}(t)}{m_{ij}(t-1)+1} & \text{if } a_{ij}(t) \neq 0, \\ \widehat{p}_{ij}(t-1) & \text{otherwise.} \end{cases} \tag{4.12}$$

$$m_{ij}(t) = \begin{cases} m_{ij}(t-1)+1 & \text{if } a_{ij}(t) \neq 0, \\ m_{ij}(t-1) & \text{otherwise.} \end{cases} \tag{4.13}$$

Theorem 2. *Assume w_j to be homogeneous across users for a sufficient amount of time, UPL provides a bounded regret given by:*

$$\mathcal{R}(t) \leq \left[\frac{4a_{max}^2 N^3 (N+1)\ln(t)}{(\Delta_{min})^2} + \frac{\pi^2}{3}N^2 + N\right]\Delta_{max}, \tag{4.14}$$

where, $a_{max} = \max_{\mathbf{A}\in\mathcal{F}} \max_{i,j} a_{ij}$*,* $\Delta_{min} = \min_{\mathbf{R_A}<\mathbf{R}^*} (\mathbf{R}^* - \mathbf{R_A})$ *and* $\Delta_{max} = \max_{\mathbf{R_A}<\mathbf{R}^*} (\mathbf{R}^* - \mathbf{R_A})$ *are the minimum and maximum difference to the reward obtained with perfect knowledge of the users' preferences, respectively.*

Proof. The proof is obtained following Theorem 2 of (Gai et al., 2012). □

4.5 Experimental Results

This section evaluates performance of the proposed approach against the approach proposed in (Zhu et al., 2013).

4.5.1 Experimental setting description

The simulations that will be presented in this section have been carried out on the simulation platform described in Chapter 2, which exploits a distributed environment where each producer and consumer is simulated by a process that runs on the same cluster of computers. In this environment, a module for the exchange of energy services based on Message Passing Interface has been implemented, i.e., a communication interface responsible to enable messaging over a communication channel. This simulator employed the platform introduced in (Agate et al., 2018) as a core. It is written in C++ where it exploits MPICH, a MPI library available for many UNIX like distributions and Windows OS. In particular, the reputation management system based on reinforcement learning of the user preferences, presented in Section 4.4, has been implemented in the *RMS* module seen in Chapter 2. The whole application scenario, in particular the energy exchange system, was simulated on the *Service Exchange* module of each agent of the network. In this simulator, a particular process is responsible for energy prediction as well as executing the reinforcement learning UPL algorithm to send consumers recommendations. At each time step, Gurobi was used to solve the optimization problem within the UPL algorithm to pick the next action. In order to keep trace of all the energy transactions occurred, at the end of every time slot, the leading process, called previously *truth holder*, collects all the output of every transaction.

Real consumption data are obtained from the Pecan[6] dataset that contains daily aggregate energy consumption data of 53 residential buildings of different types and sizes over the course of 2014. Following the concept of VPPs, producers were modeled as follows. Let us consider two groups of solar energy producers located in Lexington, Kentucky. Producers are equipped with Photovoltaic (PV) generation capabilities. The first group consists of 8 producers with 8 kW plant size and the second group is 12 producers equipped with 4 kW plant size. Furthermore, the NREL's PVWatts Calculator of the U.S. Department of Energy were used to generate the energy production over time given the solar irradiance in Lexington and the size of the PV plants. Under this assumption, in the experiments m has been set to $m = 2$. Finally, user preferences and losses were uniformly assigned at random in the sets $\{0.6, 0.8, 1\}$ and $\{1\%, 2\%, 3\%, 4\%\}$, respectively.

4.5.2 Comparison approaches

The proposed approach is compared to the algorithm proposed in (Zhu et al., 2013). This algorithm matches producers and consumers in order to minimize the transmission loss. In this method, first consumers are sorted in descending order based on the quantity of energy needed. Then, the algorithm follows such order and matches the consumers' demand with the available producers by giving precedence to those that provide the minimum loss. The experiments refer to this algorithm as "Zhu", and the reader is referred to (Zhu et al., 2013) for more details.

Since the original Zhu algorithm does not take into account the consumers' preferences, the proposed algorithm is compared with two modified versions of the original approach. In both versions, the matching criteria based on loss is replaced with the consumers' preferences to maximize the likelihood that the recommendation is accepted. Specifically, the system follows the sorted order of consumers and match each consumer c_j with the producer b_i that has the highest value of p_{ij}. Let us define an approach, called "Zhu$_P$", in which the consumer preference probabilities are perfectly known, as well as an approach called "Zhu$_{PL}$" that learns these probabilities over time by keeping track of the historic average of accepted/rejected recommendations.

[6]Pecan street inc. dataport 2014. URL www.pecanstreet.org.

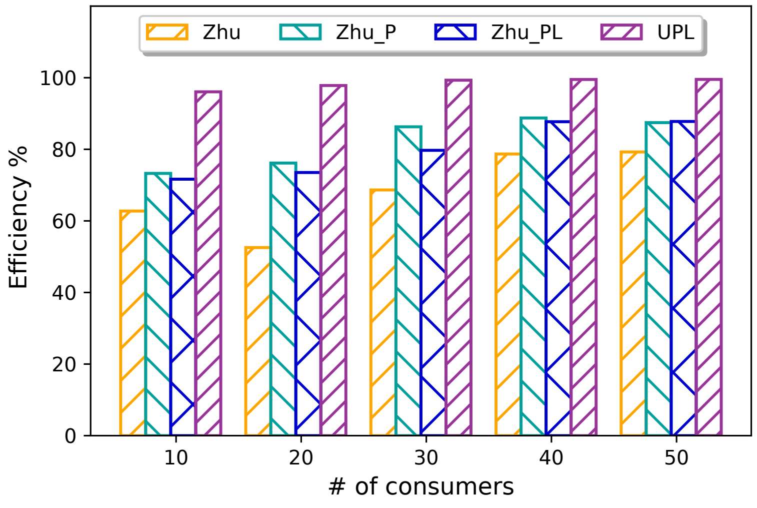


Figure 4.3: Efficiency of the algorithms with perfect knowledge of the power consumption.

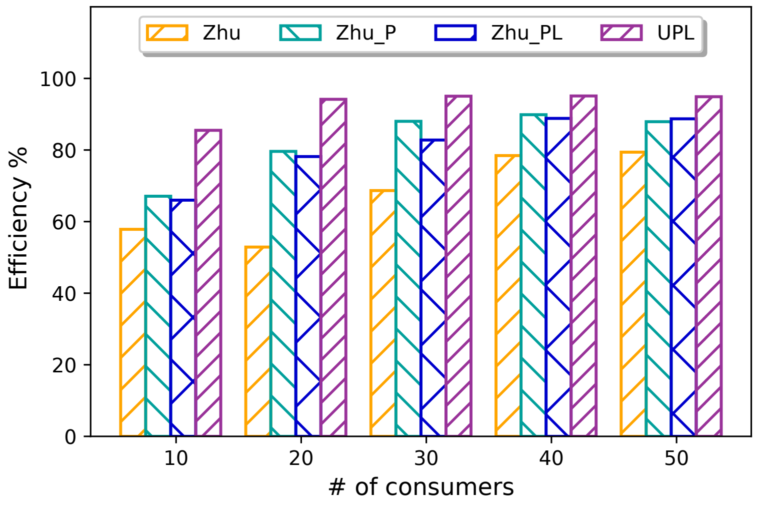


Figure 4.4: Efficiency of the algorithms with prediction of the power consumption.

4.5.3 Performance Evaluation

The proposed system is evaluated considering three experimental scenarios, as described in the following.

Experimental Scenario 1. In this scenario the efficiency of the system, is studied, defined as the ratio of actual exchanged energy over the optimum value ob-

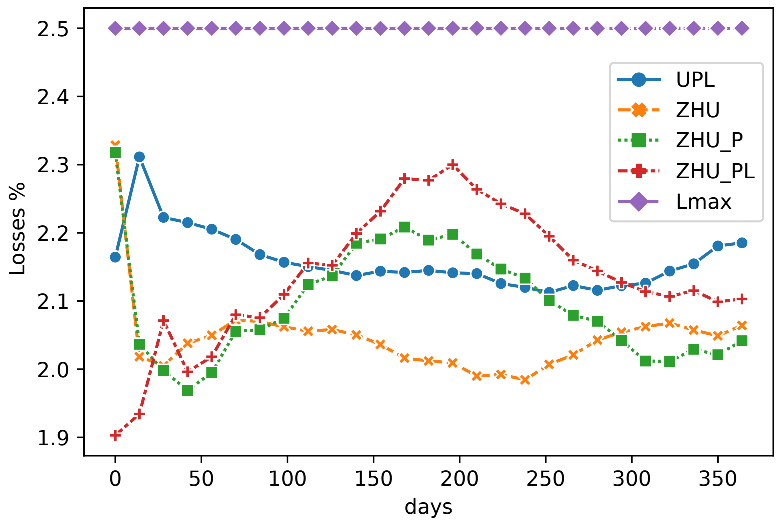


Figure 4.5: Losses incurred for the algorithms with perfect knowledge of the power consumption.

tained by solving the optimization problem in Eqs. (4.1)-(4.7) with perfect knowledge of the user behavior, each day. The efficiency is averaged over a period of a year. Figs. 4.3 and 4.4 show the result for perfect knowledge of the consumer consumption and for consumption predicted through an Exponentially Weighted Moving-Average (EWMA) with $\alpha = 0.5$ (Kansal et al., 2007), respectively.

It can be observed that including user-preference in the matching of energy resources improves the efficiency of the sharing system. As a matter of fact, all algorithms which take consumers' preferences into account, i.e., UPL, Zhu$_P$, and Zhu$_{PL}$, outperform the Zhu algorithm that just focuses on losses. As expected, Zhu$_P$ usually performs better than Zhu$_{PL}$, since it has perfect knowledge of the consumers' preferences. However, as number of the consumers increases, which is equivalent to growing number of transactions, the learning method of Zhu$_{PL}$ improves.

Overall, UPL significantly outperforms all versions of the Zhu approaches, both with perfect and estimated required energy. In fact, it achieves an efficiency close to 100% even when only the prediction of energy consumption is available, as shown in Fig. 4.4.

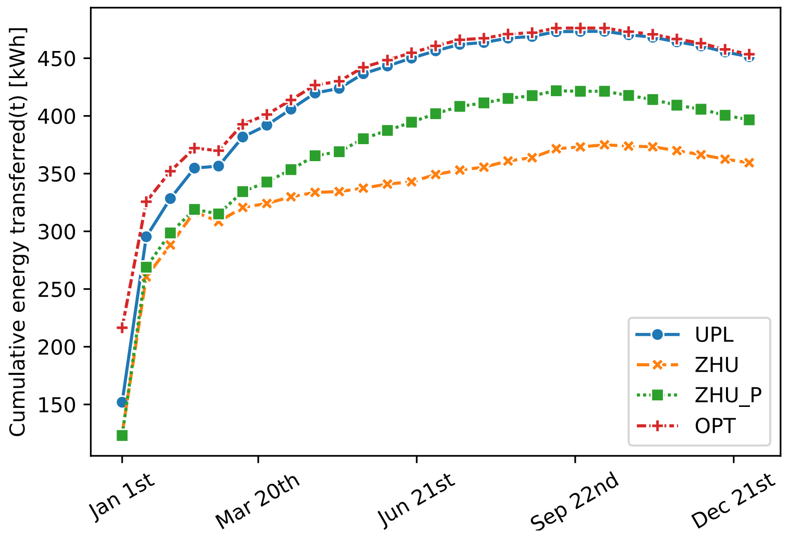


Figure 4.6: Cumulative transferred energy divided by time for different algorithms in the course of a year.

Fig. 4.5 complements the previous results by showing the percentage of energy loss in the case of perfect knowledge of energy consumption.The case of predicted energy is omitted, since similar trends were observed. L_{max} is set to 2.5%, therefore UPL does not allow exchanges between users having a percentage loss higher than this threshold. As expected, Zhu incurs the lowest loss, since this method explicitly focuses on optimizing this parameter. However, this comes at the expense of a lower efficiency as previously discussed. UPL incurs losses which are well below the L_{max} threshold, showing the efficacy of the proposed approach. Note that, there is a period during Fall/Winter in which Zhu$_P$ has lower loss than Zhu. This is due to the lower amount of energy transactions under Zhu$_P$, which results in lower losses.

Experimental Scenario 2. In this scenario the trend of energy transferred over time is studied. To this purpose, for each time t, a metric calculated as the cumulative energy transferred in the system until then, divided by t, is defined. Results are shown in Fig. 4.6. Since realistic energy production data from PVWattCalculator[7] are used, there is a seasonal effect that justifies the non-monotonic trend of

[7]SolarR Resource Data. URL https://pvwatts.nrel.gov/pvwatts.php.

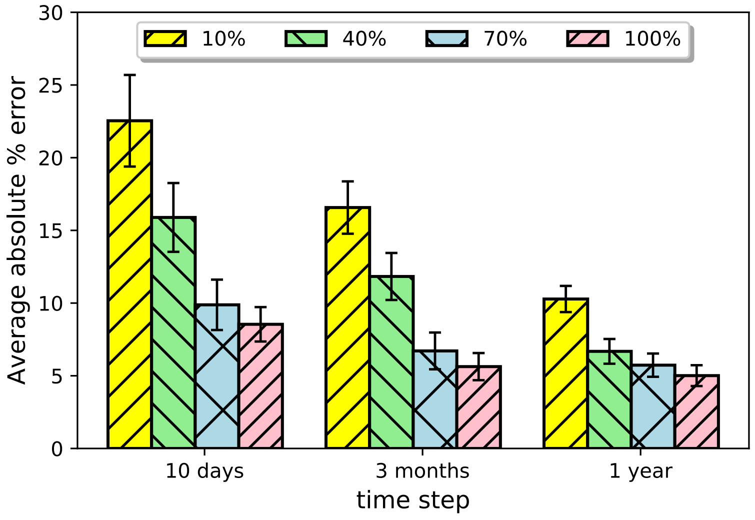


Figure 4.7: Average absolute percentage error on learned probabilities after different periods, by varying the amount of available energy in the system.

all approaches. In fact, during the Fall/Winter months there is a decrease in energy production of solar panels, which implies a decrease in the exchanged energy. Nevertheless, as Fig. 4.6 shows, UPL outperforms all approaches, demonstrating outstanding performance with negligible gap with respect to the optimal solution.
Experimental Scenario 3. In the last experimental scenario, the impact of the ratio of the produced energy on the system performance is analyzed. Intuitively, when less energy is produced, less exchanges are possible. This results in a slower learning phase, which may impact the efficiency of the system. In these experiments the produced energy is altered to be a given percentage (10%, 40%, 70%, and 100%) of the energy needed by the consumers, and investigated the system performance after 10 days, 3 months, and 1 year.

Fig. 4.7, shows the average percentage absolute error in learning the consumers' preferences, i.e., the probabilities p_{ij}. As expected, the error decreases as the amount of energy increases, as well as with time. It is worth noting that, even under just 10 days, the error is below 10% if at least 70% of the required energy is available. Interestingly, the error never reaches zero, and it tends to stabilize around 5%. This is due to the nature of reinforcement learning at the basis of UPL, which prefers exploitation versus exploration, once sufficient knowledge is

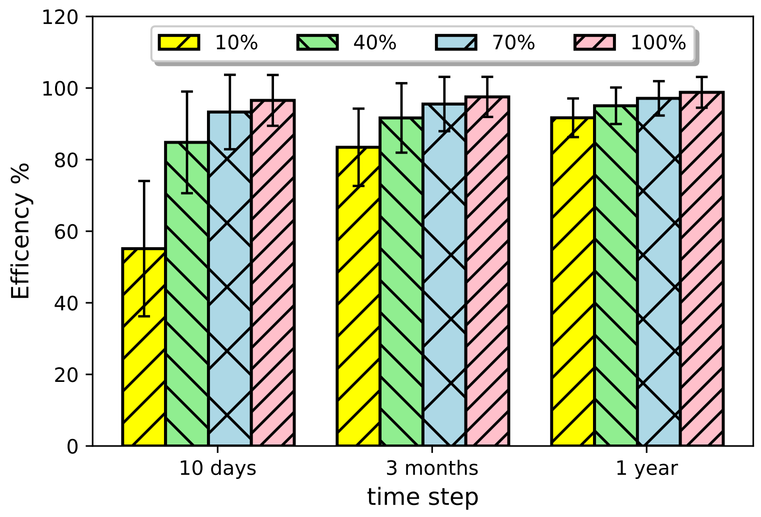


Figure 4.8: Efficiency of the system after different periods, by varying the amount of available energy in the system.

acquired. In fact, once the best consumers (i.e., those with higher chances of accepting a recommendation) are identified, these are selected more often, in order to maximize the system performance. As a result, other consumers' preferences are not learned exactly. Nevertheless, this does not prevent the system from achieving high efficiency. In fact, as shown in Fig. 4.8, the system is able to achieve more than 85% efficiency after three months of operation, under all energy availability scenarios.

Conclusions

Distributed environments where autonomous agents act cooperatively need a Reputation Management System to estimate the reliability of unknown agents before starting a service exchange. In fact, in a distributed system agents often have to interact without having a prior mutual knowledge and without having sufficient information to establish secure exchanges. Ensuring a secure and efficient collaboration between network participants leads researchers to design an RMS to estimate the reputation of agents involved in interactions and, their solutions are as good as the reputations better reflect the behavior patterns of participants. A first concern of the designers is therefore the realization of an RMS that takes into account specific accuracy requirements.

In addition, this goal leads them to worry about the application scenario on which to model their solution and many of the efforts must therefore be engaged in the creation of a simulation that can verify the correctness of their RMS and its applicability to the context of simulation. This nullifies every time the efforts of the researchers who contrarily could address their efforts towards the modeling of the RMSs.

Even more, because of their distributed nature, most RMSs are sensitive to a number of security attacks designed to exploit specific vulnerabilities. Many attackers exploit the vulnerabilities of such systems to abuse the shared resources. The main vulnerabilities come from internal attackers, who have obtained the right to participate in the sharing system, but who in different ways want to cause damage to the system by spreading, for example, false reputation information. By changing perspective and focusing on the service exchange, it is possible to notice that some agents are willing to become bad service providers and instead receive resources in an inappropriate way.

On the basis of knowledge acquired, a comprehensive framework to allow developers to assess their RMSs since the design phase, capable of evaluating its vulnerability to security attacks, easily customizable, and supporting large-scale simulation is still missing. To meet this challenge, this dissertation presented DRESS, a Distributed RMS Evaluation Simulation Software.

In the design of the simulator, the theoretical concepts of the distributed algorithms were used to model the RMS. The adoption of this model has been reflected in the use of the paradigm of communication between processes through MPI, a communication protocol for message passing.

The architectural model of the simulation framework takes advantage of the intuition to associate each agent of the reputation system to a process. This feature makes it suitable to run both on small hardware architectures and on clusters of networked computers, enabling large-scale simulations.

DRESS allows researchers to achieve several tasks relying on a set of core functionalities that can be immediately used, or extended to fit almost any need for personalization. For example, the user can address custom application scenarios, define new distributed services and generate ad-hoc agent models.

The simulation platform has been designed and developed also to give the possibility to choose different RMSs, or different policies, which should be applied to the defined context in order to immediately observe the response of the distributed application in terms of performance improvement. For example, in a resource sharing system in which the focus is on maximizing the resources exchanged, DRESS allows to immediately verify the validity of the choices made by the designers by simply visualizing the trends of the agents' reputations or the gain in terms of the actual utility perceived by the participants.

The evaluation of the RMS against a set of different attacks can be easily performed by means of some vulnerability metrics that show how different design choices impact on the robustness of the reputation system.

In order to prove the effectiveness of the proposed solution, this thesis has described how DRESS can be configured to evaluate a simple RMS that is victim of four different security attacks.

In this thesis it was shown how the designer can redefine the reputation management model by intervening in the functionalities offered by the simulator. Finally, the framework showed good capabilities in terms of scalability, demonstrating the possibility of being run on a computer cluster, reducing the simulation time.

A deeper analysis of the scalability of the simulation framework, in terms of memory occupation, computational complexity and communication, could be carried out as future research work, using advanced techniques for efficient process allocation.

In addition, the suitability of DRESS to simplify the construction of a RMS on a specific real application has been proven. In particular, this dissertation addressed a typical problem in the context of smartgrids by trying to implement an energy sharing system(ESS) under the concept of virtual power plants (VPPs). Differently from previous works, last part of this dissertation has been focused on learning realistic consumers' preferences in the optimization of ESS. A realistic user behavioral model in which users may accept, reject or ignore each producer listed in a recommendation has been considered. This model is learned through a reinforcement learning approach at the basis of the User Preference Learning (UPL) algorithm. User preferences were seen as the reputation of users and the information obtained was used to select only the most cooperative users in energy transactions in order to maximize energy exchanged. Compared to state-of-the-art approaches, experimental results based on realistic data show that the proposed approach achieves higher efficiency with comparable energy transfer losses.

As future work it would be interesting to investigate the performance of DRESS while varying the type of computational resources adopted, since not all researchers may have access to computer clusters. Another aspect worth considering is studying how to adapt DRESS to a cloud based computing platform. To achieve this task it is possible to exploit simulation-as-a-service techniques and to provide a user-friendly web interface.

Bibliography

T. Ackermann, G. Andersson, and L. Söder. Distributed generation: a definition. *Electric Power Systems Research*, 57(3):195 – 204, 2001.

V. Agate, A. De Paola, G. Lo Re, and M. Morana. A simulation framework for evaluating distributed reputation management systems. In *Distributed Computing and Artificial Intelligence, 13th International Conference*, pages 247–254, Cham, 2016a. Springer International Publishing.

V. Agate, A. De Paola, G. Lo Re, and M. Morana. Vulnerability Evaluation of Distributed Reputation Management Systems. In *InfQ 2016 - New Frontiers in Quantitative Methods in Informatics*, pages 1–8, ICST, Brussels, Belgium, Belgium, 2016b. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

V. Agate, A. De Paola, G. Lo Re, and M. Morana. A platform for the evaluation of distributed reputation algorithms. In *2018 IEEE/ACM 22nd International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 1–8, Oct 2018.

S. Althaher, P. Mancarella, and J. Mutale. Automated demand response from home energy management system under dynamic pricing and power and comfort constraints. *IEEE Transactions on Smart Grid*, 6(4):1874–1883, July 2015.

P. Asmus. Microgrids, virtual power plants and our distributed energy future. *The Electricity Journal*, 23(10):72–82, 2010.

F. M. Awuor, C.-Y. Wang, and T.-C. Tsai. Motivating content sharing and trustworthiness in mobile social networks. *IEEE Access*, 6:28339–28355, 2018.

E. Ayday and F. Fekri. Iterative trust and reputation management using belief propagation. *IEEE Transactions on Dependable and Secure Computing*, 9(3): 375–386, May 2012.

S. Ba and P. A. Pavlou. Evidence of the effect of trust building technology in electronic markets: Price premiums and buyer behavior. *MIS quarterly*, pages 243–268, 2002.

S. Bhattacharjee, A. Thakur, S. Silvestri, and S. K. Das. Statistical security incident forensics against data falsification in smart grid advanced metering infrastructure. In *ACM CODASPY*, pages 35–45, 2017.

P. Chandrasekaran and B. Esfandiari. Toward a testbed for evaluating computational trust models: experiments and analysis. *J. of Trust Management*, 2(1): 1–27, 2015.

T. F. E. R. Commission. *Reports on Demand Response & Advanced Metering*, December 17, 2015. `http://www.ferc.gov/industries/electric/indus-act/demand-response/dem-res-adv-metering.asp`.

D. Contu, E. Strazzera, and S. Mourato. Modeling individual preferences for energy sources: The case of iv generation nuclear energy in italy. *Ecological Economics*, 127:37–58, 2016.

C. Crapanzano, F. Milazzo, A. De Paola, and G. Lo Re. Reputation management for distributed service-oriented architectures. In *Fourth IEEE Int. Conf. on Self-Adaptive and Self-Organizing Systems Workshop (SASOW)*, pages 160–165, 2010.

CVSS. Common vulnerability scoring system v3.0. `https://www.first.org/cvss`, 2015.

A. De Paola and A. Tamburo. Reputation Management in Distributed Systems. In *3rd Int. Symp. on Communications, Control and Signal Processing (ISCCSP)*, pages 666–670, 2008.

M. A. Delmas, M. Fischlein, and O. I. Asensio. Information strategies and energy conservation behavior: A meta-analysis of experimental studies from 1975 to 2012. *Energy Policy*, 61:729–739, 2013.

V. Dolve, C. Jackson, S. Silvestri, D. Baker, and A. DePaola. Social-behavioral aware optimization of energy consumption in smart homes. In *IEEE DCOSS*, 2018.

R. Earle and A. Faruqui. Toward a new paradigm for valuing demand response. *The Electricity Journal*, 19(4):21–31, 2006.

M. M. Esfahani, A. Hariri, and O. A. Mohammed. A multiagent-based game-theoretic and optimization approach for market operation of multimicrogrid systems. *IEEE Transactions on Industrial Informatics*, 15(1):280–292, Jan 2019.

M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica. Free-riding and white-washing in peer-to-peer systems. In *ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, pages 228–236, 2004.

L. Fleischer, M. X. Goemans, V. S. Mirrokni, and M. Sviridenko. Tight approximation algorithms for maximum general assignment problems. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 611–620. Society for Industrial and Applied Mathematics, 2006.

K. K. Fullam, T. B. Klos, G. Muller, J. Sabater, A. Schlosser, Z. Topol, K. S. Barber, J. S. Rosenschein, L. Vercouter, and M. Voss. A specification of the agent reputation and trust (ART) testbed: experimentation and competition for trust in agent societies. In *4th Int. joint Conf. on Autonomous agents and multiagent systems*, pages 512–518, 2005.

Y. Gai, B. Krishnamachari, and R. Jain. Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations. *IEEE/ACM Transactions on Networking*, 20(5):1466–1478, Oct 2012.

F. Hendrikx, K. Bubendorfer, and R. Chard. Reputation systems: A survey and taxonomy. *Journal of Parallel and Distributed Computing*, 75:184 – 197, 2015.

J. Herkert and T. Kostyk. Societal implications of the smart grid: Challenges for engineering. In *Engineering Identities, Epistemologies and Values*, pages 287–306. Springer, 2015.

L. Hernández, C. Baladron, J. M. Aguiar, B. Carro, A. Sanchez-Esguevillas, J. Lloret, D. Chinarro, J. J. Gomez-Sanz, and D. Cook. A multi-agent system architecture for smart grid management and forecasting of energy demand in virtual power plants. *IEEE Communications Magazine*, 51(1):106–113, 2013.

K. Hoffman, D. Zage, and C. Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Computing Surveys (CSUR)*, 42(1):1, 2009.

A. A. Irissappane, S. Jiang, and J. Zhang. Towards a comprehensive testbed to evaluate the robustness of reputation systems against unfair rating attack. In *20th Conference on User Modeling, Adaptation, and Personalization (UMAP)*, pages 1–12, 2012.

D. Jelenc, R. Hermoso, J. Sabater-Mir, and D. Trček. Decision making matters: A better way to evaluate trust models. *Knowledge-Based Systems*, 52:147–164, 2013.

K. Jhala, B. Natarajan, and A. Pahwa. Prospect theory based active consumer behavior under variable electricity pricing. *IEEE Transactions on Smart Grid*, pages 1–1, 2018.

H. D. Johansen, R. V. Renesse, Y. Vigfusson, and D. Johansen. Fireflies: A secure and scalable membership and gossip service. *ACM Transactions on Computer Systems (TOCS)*, 33(2):5, 2015.

J. Johnson, J. Flicker, A. Castillo, C. Hansen, M. El-Khatib, D. Schoenwald, M. Smith, R. Graves, J. Henry, T. Hutchins, et al. Design and implementation of a secure virtual power plant. Technical report.

D. Kahneman. Maps of bounded rationality: Psychology for behavioral economics. *American economic review*, 93(5):1449–1475, 2003.

S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. In *12th Int. Conf. on World Wide Web*, pages 640–651, 2003.

A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava. Power management in energy harvesting sensor networks. *ACM Transactions on Embedded Computing Systems (TECS)*, 6(4):32, 2007.

R. Kerr and R. Cohen. Treet: the trust and reputation experimentation and evaluation testbed. *Electronic Commerce Research*, 10(3):271–290, Dec 2010.

E. Koutrouli and A. Tsalgatidou. Reputation systems evaluation survey. *ACM Computing Surveys (CSUR)*, 48(3):35, 2016.

S. Lakshminarayana, T. Q. S. Quek, and H. V. Poor. Cooperation and storage tradeoffs in power grids with renewable energy resources. *IEEE Journal on Selected Areas in Communications*, 32(7):1386–1397, July 2014.

Q. Lian, Z. Zhang, M. Yang, B. Y. Zhao, Y. Dai, and X. Li. An empirical study of collusion behavior in the maze p2p file-sharing system. In *27th Int. Conf. on Distributed Computing Systems (ICDCS'07)*, pages 56–56, 2007.

N. Liu, X. Yu, C. Wang, C. Li, L. Ma, and J. Lei. Energy-sharing model with price-based demand response for microgrids of peer-to-peer prosumers. *IEEE Transactions on Power Systems*, 32(5):3569–3583, Sep. 2017.

N. A. Lynch. *Distributed algorithms.* Morgan Kaufmann, 1996.

F. G. Mármol and G. M. Pérez. Trmsim-wsn, trust and reputation models simulator for wireless sensor networks. In *IEEE International Conference on Communications (ICC'09)*, pages 1–5, 2009.

S. Marti and H. Garcia-Molina. Taxonomy of trust: Categorizing p2p reputation systems. *Computer Networks*, 50(4):472–484, 2006.

O. Palizban, K. Kauhaniemi, and J. M. Guerrero. Microgrids in active network management part i: Hierarchical control, energy storage, virtual power plants, and market participation. *Renewable and Sustainable Energy Reviews*, 36:428–439, 2014.

Y. Parag and B. Sovacool. Electricity market design for the prosumer era. *Nature Energy*, 1:16032, March 2016.

M. H. Rehmani, M. Reisslein, A. Rachedi, M. Erol-Kantarci, and M. Radenkovic. Integrating renewable energy resources into the smart grid: Recent developments in information and communication technologies. *IEEE Transactions on Industrial Informatics*, 14(7):2814–2825, 2018.

E. Ropuszyńska-Surma and M. Węglarz. Profiling end user of renewable energy sources among residential consumers in poland. *Sustainability*, 10(12):4452, 2018.

A. Salehi-Abari and T. White. Dart: a distributed analysis of reputation and trust framework. *Computational Intelligence*, 28(4):642–682, 2012.

S. Silvestri, D. A. Baker, and V. Dolce. Integration of social behavioral modeling for energy optimization in smart environments. In *ACM Social Sense*, pages 97–97, 2017.

F. Sioshansi. *The sorry state of demand response in the U.S.*, 2019 (accessed February 3, 2020). https://energypost.eu/the-sorry-state-of-demand-response-in-the-u-s/.

Y. Sun and Y. Liu. Security of online reputation systems: The evolution of attacks and defenses. *IEEE Signal Processing Mag.*, 29(2):87–97, 2012.

S. Tadelis. The economics of reputation and feedback systems in e-commerce marketplaces. *IEEE Internet Computing*, 20(1):12–19, 2016.

E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer. Taxonomy and survey of collaborative intrusion detection. *ACM Computing Surveys (CSUR)*, 47(4):55, 2015.

M. Vasirani, R. Kota, R. L. Cavalcante, S. Ossowski, and N. R. Jennings. An agent-based approach to virtual power plants of wind power generators and electric vehicles. *IEEE Transactions on Smart Grid*, 4(3):1314–1322, 2013.

O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad. A survey on trust and reputation models for web services: Single, composite, and communities. *Decision Support Systems*, 74:121–134, 2015.

K. Wang, X. Qi, L. Shu, D.-J. Deng, and J. J. Rodrigues. Toward trustworthy crowdsourcing in the social internet of things. *IEEE Wireless Communications*, 23(5):30–36, 2016.

K. Wang, X. Hu, H. Li, P. Li, D. Zeng, and S. Guo. A survey on energy internet communications for sustainability. *IEEE Transactions on Sustainable Computing*, 2(3):231–254, July 2017.

A. G. West, S. Kannan, I. Lee, and O. Sokolsky. An evaluation framework for reputation management systems. *Trust Modeling and Management in Digital Environments: From Social Concept to System Development*, pages 282–308, 2010.

Y. Zhang, W. Wang, and S. Lü. Simulating trust overlay in p2p networks. *Computational Science–ICCS 2007*, pages 632–639, 2007.

T. Zhu, A. Mishra, D. Irwin, N. Sharma, P. Shenoy, and D. Towsley. The case for efficient renewable energy management in smart homes. In *Proceedings of the Third ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 67–72. ACM, 2011.

T. Zhu, Z. Huang, A. Sharma, J. Su, D. Irwin, A. Mishra, D. Menasche, and P. Shenoy. Sharing renewable energy in smart microgrids. In *2013 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, pages 219–228, April 2013.