UNIVERSITATEA
BABEŞ-BOLYAI
Cluj-Napoca

FACULTATEA DE ŞTIINŢE
ECONOMICE ŞI GESTIUNEA
AFACERILOR

România
Ministerul Educaţiei, Cercetării, Tineretului şi Sportului
Universitatea Babeş-Bolyai
Cluj-Napoca
Facultatea de Ştiinţe Economice
şi Gestiunea Afacerilor
Str. Teodor Mihali nr. 58-60
400591, Cluj-Napoca
Tel: 0264 418655
Fax: 0264 412570
E-mail: econ@econ.ubbcluj.ro

Babeş-Bolyai University
Business Information Systems Department

Habilitation Thesis

Candidate:
Gheorghe Cosmin SILAGHI

Cluj-Napoca
2012

Abstract

Peer-to-Peer systems represent one of the most dynamic and challenging research area of distributed computing. With theoretical grounds on multi-agent systems, P2P systems get closer to the practice, allowing large communities to automate collaboration and deliver various services at low costs and good quality. Agents are defined as computer systems capable of independent and autonomous action on behalf of their users or owners, figuring out what need to be done to satisfy the design objectives. The definition leads the research towards formal aspects of the intelligence, agent research community being interested on formalizing the interaction, modeling agent behaviors, devising intelligent algorithms and approaching pro-activity. By contrast, in distributed computing a peer represents just another node with the same rank with others. Peers interaction is practically modeled using the basic client-server paradigm, a peer playing either the role of the client, either the role of the server, without the requirement that some master (central) peer to coordinate everything. P2P systems gained such popularity that they emerged to deliver various services. Initially, P2P communities gathered together to distribute music; later they formed huge distributed file systems where everyone can store and find various items from books, movies and others. Nowadays, P2P systems represent the next step of the grid concept evolution, peers delivering sophisticated services besides storage and computing power. Desktop grids under the volunteer computing paradigm fit exactly this latter approach, home Internet users donating their computing cycles to world-wide distributed scientific projects.

In this context, one can note that distributed computing research community, besides focusing on the practical implementation aspects of P2P systems, adopt

more and more ideas originating in the agent research field of artificial intelligence. Peers are smarter and smarter, systems are delivering sophisticated services, moving apart from the classical storage and computing power capabilities. Everything become transparent under the umbrella of cloud computing, where platforms, infrastructures and services are virtualized and delivered on-demand to interested customers. Of course, with their increased usage and sophistication of the delivered services, various research problems sprinkled out, from the classical resource management problem towards how to face with malicious behaviors of peers who are interested to benefit from the community without contributing.

If peers are autonomous and independent each of another, it came out that they might exhibit self-interested behaviors, including undesired ones. Applied to a computer system, dependability is defined as the trustworthiness of a computing system which allows reliance to be justifiably placed on the service it delivers. To trust and rely systems and services offered on the Internet by anonymous contributors, measures and controls must be in place. Contributing to such systems shall assure the participants that they will get a fair value back, meaning that proper designed resource management mechanisms should exist. Thus, there is a large space to use game theoretical analysis for proper design of such service-based P2P systems.

This thesis summarizes our main scientific contribution, spread over the last years, after the completion of the PhD. While during our PhD research we approached and investigated tools towards automated collaboration inside agent communities; with this research we stepped in the field of distributed computing, approaching resource management in systems composed of anonymous contributors. Within this thesis, we centered our contribution on systems with open participation, which transparently deliver services under the umbrella of Service Level Agreements. Assuming that a P2P system deployed over Internet delivers some services and the system allows everyone (with various capabilities) to contribute, we devised a reputation model for resource usage control and access control in the system. Given that contributors are self-interested and such a system is organized as bargaining marketplace, we devised a bilateral negotiation protocol towards service level agreements under time constraints. Approaching

the challenging field of volunteer computing and desktop grids, we were firstly to define, model and propose a solution against colluding attackers.

Our contribution enlarges the developments of grid, P2P and service systems economics. We took principles and concepts developed, formalized and analyzed part of the economic theory and apply them in automatic systems. The main characteristic is that our proposed methods are fully automated and let the systems using them to fulfill their requirements without the human intervention. When assessing the trust, our reputation model is based on the concept of utility. Actors operating the service systems nodes need to define their goals and let the utility reputation model to assess the others based on those goals. The automatic SLA negotiation is strongly based on the strategic bargaining theory, strongly encouraged by the definition adopted in the WS-Agreement-Negotiation protocol. Peers are equipped with learning techniques and negotiate self-interested, given that they are able to express their preferences over the negotiation solutions. When modeling the colluding malicious behavior in desktop grids, nodes actions are observed on a time frame and a statistical model is computed for each node. This statistical model allows the definition of a clustering procedure, to extract out the majority of nodes that are behaving honestly.

The content of the thesis is disseminated in prestigious world journals and was presented at recognized conferences over Europe and USA. The reputation model was firstly presented at the 1st CoreGrid symposium, a premiere European event on Grid Computing for the dissemination of the results from European and member states initiatives as well as other international projects in Grid research and technologies. Up to now, this model attracted 11 citations in ISI WoS and Scopus. The review of the reputation model was initially published as a CoreGrid technical report and attracted 23 citations in ISI WoS and Scopus. In August 2008, at the end of the CoreGrid FP6 project, it was ranked in the top 10 out of 178 technical reports of CoreGrid by the number of downloads. The journal version published in Security and Communication Networks at Wiley includes as an addition the implementation of the model in the FP6 GridTrust project and experimentation with access control. The sabotage tolerance model against collusion in desktop grids was firstly presented at the PCGrid 2008 workshop in conjunction with the prestigious IEEE IPDPS conference and next, it was

accepted for publication in Journal of Grid Computing at Springer. The model attracted follow-ups, mentioning here the PhD thesis of E. Staab, defended in 2010 at University of Luxembourg and works done at INRIA Grenoble by L.C. Canon. We further contributed to the development of the Maximum Independent Set approach of colleagues from University of Coimbra, who got published in the Journal of Parallel and Distributed Computing. The SLA negotiation model under time constraints was entirely developed at Babes-Bolyai University of Cluj-Napoca. Initially, the model was presented at the 7th International Workshop on the Economics and Business of Grids, Clouds, Systems, and Services GECON 2010 and the journal publication is with Future Generation Computer Systems journal at Elsevier.

Contents

List of Figures

List of Tables

Chapter 1

Dependable Resource Management Tools towards Automated Collaboration in Heterogeneous Computing Environments

1.1 Introduction

Peer-to-Peer systems represent one of the most dynamic and challenging research area of distributed computing. With theoretical grounds on multi-agent systems, P2P systems get closer to the practice, allowing large communities to automate collaboration and deliver various services at low costs and good quality.

Agents are defined as computer systems capable of independent (autonomous) action on behalf of their users or owners, figuring out what need to be done to satisfy the design objectives, rather than constantly being told [Wooldridge, 2009]. The definition leads the research towards formal aspects of the intelligence, agent research community being interested on formalizing the interaction, modeling agent behaviors, devising intelligent algorithms and approaching pro-activity. By contrast, in distributed computing a peer represents just another node with the same rank with others. Peers interaction is practically modeled using the basic client-server paradigm, a peer playing either the role of the client, either the role of the server, without the requirement that some master (central) peer to coordinate everything. P2P systems gained such popularity that they emerged to deliver various services. Initially, P2P communities gathered together to distribute music; later they formed huge distributed file systems where everyone can store and find various items from books, movies and others. Nowadays, P2P systems represent the next step of the grid concept evolution [Foster and Iamnitchi, 2003], peers delivering computing power besides storage. Desktop grids under the volunteer computing paradigm fit exactly this latter approach, home Internet users donating their computing cycles to world-wide distributed scientific projects.

In this context, one can note that distributed computing research community, besides focusing on the practical implementation aspects of P2P systems, adopt more and more ideas originating in the agent research field of artificial intelligence. Peers are smarter and smarter, systems are delivering sophisticated services, moving apart from the classical storage and computing power capabilities. Everything become transparent under the umbrella of cloud computing, where platforms, infrastructures and services are virtualized and delivered on-demand to interested customers. Of course, with their increased usage and sophistication

of the delivered services, various research problems sprinkled out, from the classical resource management problem towards how to face with malicious behaviors of peers who are interested to benefit from the community without contributing.

If peers are autonomous and independent each of another, it came out that they might exhibit self-interested behaviors, including undesired ones. Applied to a computer system, *dependability* is defined as *the trustworthiness of a computing system which allows reliance to be justifiably placed on the service it delivers* IFIP Working Group 10.4 [1988]. To trust and rely systems and services offered on the Internet by anonymous contributors, measures and controls must be in place. Contributing to such systems shall assure the participants that they will get a fair value back, thus proper designed resource management mechanisms should exist. Thus, there is a large space to use game theoretical analysis for proper design of such service-based P2P systems.

This thesis summarizes our main scientific contribution, spread over the last years, after the completion of the PhD. While during our PhD research we approached and investigated tools towards automated collaboration inside agent communities; with this research we stepped in the field of distributed computing, approaching resource management in systems composed of anonymous contributors. We centered our contribution around systems with open participation, that transparently deliver services under the umbrella of Service Level Agreements [Andrieux et al., 2007]. Assuming that a P2P system deployed over Internet delivers some services and the system allows everyone (with various capabilities) to contribute, we devised a reputation model for resource usage control and access control in the system. Given that contributors are self interested and such a system is organized as bargaining marketplace, we devised a bilateral negotiation protocol towards service level agreements under time constraints. Approaching the challenging field of volunteer computing and desktop grids, we were firstly to define, model and propose a solution against colluding attackers.

Our contribution enlarges the developments of grid, P2P and service systems economics. We took principles and concepts developed, formalized and analyzed part of the economic theory and apply them in automatic systems. The main characteristic is that our proposed methods are fully automated and let the systems using them to fulfill their requirements without the human intervention.

When assessing the trust, our reputation model is based on the concept of utility. Actors operating the service systems nodes need to define their goals and let the utility reputation model to assess the others based on those goals. The automatic SLA negotiation is strongly based on the strategic bargaining theory, strongly encouraged by the definition adopted in the WS-Agreement-Negotiation protocol [Waeldrich et al., 2011]. Peers are equipped with a learning techniques and negotiate self-interested, given that they are able to express their preferences over the negotiation solutions. When modeling the colluding malicious behavior in desktop grids, nodes actions are observed on a time frame and a statistical model is computed for each node. This statistical model allows the definition of a clustering procedure, to extract out the majority of nodes that are behaving honestly.

The content of the thesis is disseminated in prestigious world journals and was presented at recognized conferences over Europe and USA. The reputation model [Silaghi et al., 2007b] was firstly presented at the 1st CoreGrid symposium, a premiere European event on Grid Computing for the dissemination of the results from European and member states initiatives as well as other international projects in Grid research and technologies. Up to now, this model attracted 11 citations in ISI WoS and Scopus. The review of the reputation model [Silaghi et al., 2007a] was initially published as a CoreGrid technical report and attracted 23 citations in ISI WoS and Scopus. In August 2008, at the end of the CoreGrid FP6 project, it was ranked in the top 10 out of 178 technical reports of CoreGrid by the number of downloads. The journal version [Arenas et al., 2010] published in Security and Communication Networks at Wiley includes as an addition the implementation of the model in the FP6 GridTrust project and experimentation with access control. The sabotage tolerance model against collusion in desktop grids [Silaghi et al., 2008a] was firstly presented at the PCGrid 2008 workshop in conjunction with the prestigious IEEE IPDPS conference and next, it was accepted for publication in Journal of Grid Computing at Springer [Silaghi et al., 2009]. The model attracted follow-ups, mentioning here the PhD thesis of E. Staab, defended in 2010 at University of Luxembourg (citation in [Staab and Engel, 2009]) and works done at INRIA Grenoble by Canon et al. [2010, 2011].

We further contributed to the development of the Maximum Independent Set approach of colleagues from University of Coimbra, who got published in the Journal of Parallel and Distributed Computing [Araujo et al., 2011]. The SLA negotiation model under time constraints was entirely developed at Babeş-Bolyai University of Cluj-Napoca. Initially [Silaghi et al., 2010], the model was presented at the 7th International Workshop on the Economics and Business of Grids, Clouds, Systems, and Services GECON 2010 and the journal publication [Silaghi et al., 2011] is with Future Generation Computer Systems journal at Elsevier.

This thesis develops as follows. In section 1.2 we define the concepts we are dealing with, making a short introduction to P2P systems and their research problems. Section 1.3 investigates reputation models proposed for grid computing and propose an original utility-based reputation model to overcome the problem of the subjective feedback. Section 1.4 defines the collusion sabotage problem of the desktop grids and propose a statistical tool to fight against collusion. With its follow-ups, the devised model tackle well both the Sybil attack and the whitewashers, keeping the model assumptions to fit the practical realities of the desktop grids. Section 1.5 presents a time constrained negotiation protocol to be applied for service level values negotiation in bilateral bargaining, with self-interested peers. Each section first introduces its research question, next, it presents some background conceptual definitions and the state-of-the-art related work in the field and in its central part presents the scientific contribution validated with experimentation and discussed against the research objectives and state-of-the-art.

1.2 Background and Related Work

In this section we will introduce the main concepts used and developed part of our scientific contribution. We will generally introduce P2P systems, as distributed computer systems capable of delivering some services based on volunteer contribution of participants. Next, we will introduce the Service Level Agreement concept, that regulates the understanding between a service consumer and a provider. Finally, we will shortly brief desktop grids, as a sort of P2P system aggregating and delivering computational resources over a network.

1.2.1 P2P systems

According to Tanenbaum and van Steen [2001], a distributed systems is *a collection of independent computers that appear to the users of the system as a single computer*. P2P systems do not have a clear definition. What is widely agreed, is that a peer is just another node with the same rank with others, and systems composed on such peers do not have centralized single point of failures. In general, P2P networks are formed as overlays on the Internet, being endowed with several typical features like self-organization, distributed control and node equivalence. Node equivalence means that each component of the system has the same responsibilities, acting either as a client, either as a server, depending on the context of the interaction. In general, P2P systems play the role of resource sharing [Trunfio et al., 2007]. Accomplishing this goal, P2P systems resemble with grid systems, which aggregate huge computing and storage resources and offer them on demand to various applications.

At their beginning, P2P systems widespread on the Internet to distribute copyrighted music files. File sharing was its main purpose. Next, peers grouped together to contribute in volunteer computing projects by donating the idle time of their desktop computers. Besides file sharing and distributed processing, now, P2P systems represents the overlay to support various applications, like instant messaging, voice-over-ip, managing and sharing information, collaborative developments etc. Nowadays, we can speak about P2P service systems [Milojicic et al., 2003], where the SOA concept links together with P2P computing, services being delivered on the top of P2P overlays.

P2P systems possess some desirable properties like decentralization, scalability, cost and efficiency, pervasiveness, all of them accomplished with the intrinsic design and functioning of such systems.

When speaking about P2P systems, several issues are of great importance, all representing hot research topics:

- which is the structure of the P2P network and what mechanisms are in place to keep this structure running?
- how resources are spread over the network, and which mechanisms are in place to discover the resources?
- which principles govern the network to keep a fair balance between the contribution and consumption of peers?
- what services are delivered by the system and whether some assurance mechanisms are in place to have a warranty on (the quality of) the delivered services?
- is the P2P system vulnerable to malicious attacks and what mechanisms are in place to defend the network from manipulation and sabotage?

Below, we will shortly approach them, giving an introduction to the main concepts and developments of P2P research. For detailed surveys concerning above-mentioned issues in P2P systems, the reader can consult Ciccarelli and Lo Cigno [2011]; Meshkova et al. [2008]; Palomar et al. [2006]; Risson and Moors [2006]; Trunfio et al. [2007].

1.2.1.1 Structuring P2P systems and discovering resources

Regardless their topology, P2P systems main goal is to store some information and retrieve it in an efficient way. Given that all nodes in the system are potential (service) providers, discovering the peer responsible for some information represents the most important operation of a P2P network.

In general, P2P systems are split in two main categories: structured and unstructured P2P systems. Structured P2P systems maintain a topology of the

peers, in the sense that a clear routing procedure is followed to reach a peer from any part of the network. Besides, structured P2P systems control with strict rules how the target information is distributed to peers or how peers contribute towards the realization of the target service. Practically, each resource is identified by an id and peers in the network are made responsible of knowing where are located the resources within a subset of the id space. By contrast, unstructured P2P systems let peers to enter the network at any point, allowing unstructured graphs to be formed. There is no control regarding the peer a given resource is located.

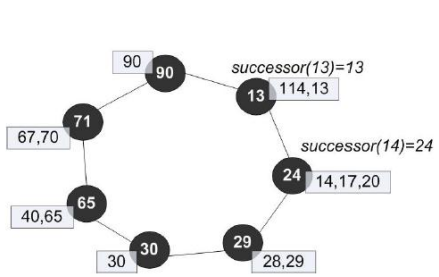
Self-organization is a strong aspect of structured P2P systems, developed protocols supplying with clear procedure about the re-organization of the network when peers leave out. Besides, we describe in short, the most important P2P topologies.

Structured P2P systems

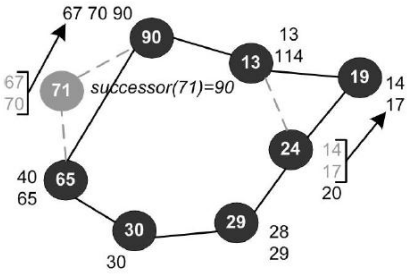
Structured P2P systems maintain the rigid interconnection between peers using the concept of *Distributed Hash Table*. DHTs are based on a hash function, which, applied both to peers and their information, determines the place of the peer in the linked network, as well the routing table for information search. In general, DHT-based systems are designed to support a search performance of up to $\mathcal{O}(\log N)$ hops to match exact queries.

Chord [Stoica et al., 2001] was the first structured P2P system. It is based on a hash function that distributes peers and their files in a m-bit key search space. Peers are organized in a double linked circled list (as in figure 1.1) and each peer stores the index of the files mapped by the hash function in a given interval of the key space. The information lookup process is similar with a binary search, delivering a performance of $\mathcal{O}(\log N)$. Chord easily integrates new incoming peers in the network. When joining, each peer is made responsible with a key space interval and all information mapping there is moved to the new peer. When a peer leaves the network, all its assigned keys are transfered to the successor peer in the topology. The division of the key space in intervals assure load balancing, as each peer is responsible for an equal number of keys, in probabilistically terms.

CAN [Ratnasamy et al., 2001] is another DHT-based P2P topology. It differs from Chord by the fact that peers are arranged in a d-dimensional torus, each being connected with its neighbor in each dimension, thus having $\mathcal{O}(d)$ neighbors.



(a) Key distribution to nodes, using successor(k) function



(b) Node 71 leaves the system and its associated keys (67 and 70) are relocated to node 90; node 19 joins the network and gets keys 14 and 17 from node 24.

Figure 1.1: Fragment of a Chord ring [Moca, 2010]

The d-dimensional key space is divided among the peers, each peer being responsible for the keys corresponding to the points in its subspace. Information search requests are forwarded among the d dimensions, and the search performance is $\mathcal{O}(N^{1/d})$. A joining peer contacts a peer already in the system who splits its search space in two parts. Leaving the network works opposite, one neighbor peer taking responsibility of the key subspace of the leaving peer.

Another approach is considered in Pastry [Rowstron and Druschel, 2001]. Each peer is mapped into a random node identifier of 128 bits. The structure of the system is organizing peers in k circles, according with their nodeId. For search routing, each peers maintain pointers towards at least one peer of each division of its nodeId space. Each lookup message is directed to the peer whose nodeId has the longest common prefix with the search key. Each peer of Pastry fingers $2k$ closest peers, thus, this scheme allows a piece of information to be located in $\log N$ steps. Tapestry [Zhao et al., 2004] builds its structure similarly with Pastry, but avoids load imbalances by publishing location pointers throughout the network in order to facilitate routing to highly requested objects with low network overhead.

Kademlia [Maymounkov and Mazieres, 2002] works somehow similar with Pastry. Each node is identified by a 160 bits key obtained by applying SHA1 hash function. Each peer has at least one neighbor, with distance between keys from $2i$ and $2i + 1$, $0 \leq i < \log N$. Each peers maintain fingers to other k peers in

each bucket and each file is replicated on k peers closest to its key. A query is routed by performing XOR on the requesting peer key and the lookup key and forwarding the lookup query to the identified bucket. Kademlia was the first P2P network to achieve world-wide scale deployment.

Other structured P2P systems are hierarchical, in the sense that they complement the DHT-based structure with some hierarchy to isolate some nodes for reasons like fault tolerance, improving bandwidth utilization, or speeding the resource location. Cyclone [Artigas et al., 2005] forms subnetwork inside the DHT, by dividing the ID of each node in two parts. Other approaches [Kaashoek and Karger, 2003; WepiwÉ and Simeonov, 2006] exploit the properties of the De-Bruijn graphs, creating co-centric additional rings in the DHT network. With this enhancement, HiPeer locates a file in a number of hops equal with the number of rings in the system.

Unstructured P2P systems

By contrast, unstructured P2P systems do not maintain a strict structure of the network. Each peer joins in at some place, being randomly connected with other peers in the network. We have seen previously, that the some structure helps either to interconnect the peers with strict networking rules, either to facilitate information retrieval. In unstructured networks, in general, no particular knowledge about information location exists, peers finding their relevant piece of information by queries flooded on the network.

Napster[1] was considered the first unstructured P2P network, although its principle is quite different. It had a central server that maintains a directory with the location of every file shared by the participating peers. This central index does not scale and, is not a characteristics of the P2P system. Kazaa[2] and Gnutella[3][Adar and Huberman, 2000] are examples of unstructured P2P networks, each peer maintaining the addresses of some neighbors. For information location, each query pass firstly to the neighbors, and, then from neighbors to neighbors flooding the network. The query is tagged with a Time-to-live (TTL)

[1]http://www.napster.com/

[2]http://www.kazaa.com/

[3]http://rfc-gnutella.sourceforge.net/

parameter, which decreases as the query passes one hop. When the TTL reaches 0, the query is not further forwarded.

The big problem of the unstructured networks is exactly this query flooding, which is very expensive. Research focused to improve this with a number of approaches. Dynamic querying[Fisk, 2003] initially launch the query only to a subset of neighbor peers, with a low TTL. Only if the requested file is not found, then the query is propagated to a bigger subset of peers, with a higher TTL. Random walk means that each query is forwarded to a randomly selected subset of the neighbors of the current peer. With this technique it is impossible to know in advance how many hops are needed to identify a resource, or even if a resource is identified. Despite this limitation, it was proved that random walk is a promising technique [Fletcher et al., 2005] to solve the search problem in unstructured P2P system. Random walk performance is heavily influenced by the topology of the unstructured network and the load of every peer. DANTE [Rodero-Merino et al., 2009] was proposed as a self-adapting P2P system that changes the topology on the fly, given the actual load of the network, such that to enhance the performance of the random walk search. Random walk can be improved by statistical information [Lv et al., 2002], or by the incorporation of the semantical information [Garcia-Molina and Crespo, 2003].

Super peer approaches

Structured P2P systems have the advantage that every resource stored by the network can be retrieved and no false negatives (a false negative response is issued when the protocol fails to identify a resource present on the system). But, the main disadvantage is that they do not encourage complex queries (e.g. based on keywords), are re-configuring the structure in the presence of high churn is costly. On contrast, unstructured P2P systems accommodate efficiently very volatile nodes, although queries can produce false negatives. Pastry and Kademlia, presented above as structured P2P exhibit some characteristics of the unsupervised P2P networks, as they present a query forwarding procedure originating in the unsupervised broadcasting. Although, because of their structuring, we classify them as supervised networks.

Super peer approaches consider some nodes exhibiting some privileged functionalities, acting as central servers for peers linked to them. In general, super

peer approaches have two layers: a layer representing an overlay of the super peers and the bottom layer with the peers linked in an unsupervised fashion to the super peers. The super peer overlay might be organized like a supervised P2P topology. Such networks exhibit the loose architecture of unsupervised topologies, while supporting partial matching queries and showing a good scalability of the system. A problem of the super peer approach is the clustering procedure adopted in order to elect the privileged peers.

A two layer architecture is proposed in Talia and Trunfio [2005] for resource discovery in OGSA compliant grids. The upper layer consists of *peer services* helping the resource discovery and *contact services* allowing *peer services* to organize themselves in a structured P2P network. Basic *index services* as supplied by the Globus-enhanced virtual organizations, are residing on the lower level and are linked to the *peer services*. A P2P grid information service is designed in Mastroianni et al. [2005]. The super peer node acts as a centralized server for some regular peers and the super peers organize themselves in a P2P overlay at the higher level. The super peer stores the metadata of the local physical organizations composed of grid nodes from some administrative domain. It is also responsible with the communication with other physical organizations. Another somehow similar super peer system is devised by Puppin et al. [2005]. Marzolla et al. [2007] devise a tree-shaped overlay topology and focuses on the routing index in order to efficiently route queries and update messages in the presence of highly variable data.

We note that the hybrid approaches arise with the emergence of complex P2P networks, peers supplying other services than information storage. Our scientific contribution will evolve around such service-based P2P networks. But, we choose not to focus on the topology; thus, in general, we will assume that our P2P systems are unsupervised organized. While network topology facilitates the resource discovery, in our research we will be interested in designing the interaction protocol between peers and the rules governing the service delivery.

1.2.1.2 Economics of P2P systems and sabotage

In general, P2P systems - regardless of their topology, are open system. Interested actors can join the network and benefit from the participation. In general, in P2P communities, there is no supervisor to control the actions happening inside the network. Further, all peers are assumed to contribute. If peers participation decreases and there is no new information of interest spread on the network, the network activity might diminish up to the network dismissal. The lack of a centralized supervision is a beneficial property of such networks, because it assures scalability. But on the other side, actions of all peers become hidden. Thus, it come out the question about how to monitor the network evolution and how to loosely impose cooperation, while each peer might behave self-interested. As we will see in this subsection, the intrinsic characteristics of P2P systems leave space for various attacks. How robust are the P2P systems to those attacks? In order to attract participants, such systems should possess some incentive rules. On one side, these economic-based rules should attract participants to the network; on the other side they should protect the participants from being exploited - thus assuring the long lasting of the system. In this subsection, we present several economic-based ideas employed in P2P systems with this respect.

The most popular P2P system organized on some economic basis is BitTorrent [Cohen, 2003]. BitTorrent addresses the problem of fairness identified in Gnutella [Adar and Huberman, 2000] as free riding, where peers join the network only to download files, without contributing. While in such P2P networks, the total download rate across all downloaders must, mathematically be equal to the total upload rate, to make peers happy, individual download rates should be proportional with the individual upload rates. In BitTorrent, this rule is achieved with the choking algorithm, peers reciprocate uploading to peers which upload to them, with the goal of, at any time, having several connections which are actively transferring in both directions. This is a sort of barter; and the widespread adoption of BitTorrent proves the solidity of this simple economic principle.

Free riding represents a usual incentive for peers. In general, contributing to the network costs, thus, there is a strong disincentive to share. We identify here the strong fundamental tension between individual rationality and collective

welfare, largely analyzed by the economic theory [Kreps, 1990]. We have seen that the BitTorrent approach to free riding is the tit-for-tat strategy. In P2P streaming applications, Chu et al. [2004] propose a taxation scheme in which resource-rich peers that contribute with a larger bandwidth subsidize for the resource-poor peers. Here, the designer is interested to keep the resource-poor peers[1] participating in the system and the taxation model redistributes the total welfare. Habib and Chuang [2004] propose a rank-based peer selection mechanism in which contributors are rewarded with flexibility and choice in peer selection. Thus, free riders receive less choice for peer selection, thus, they will not enjoy the service for which they joined the network.

A method widely approached especially in service-based P2P systems for trading multiple resource types represents the introduction of tokens or the design of a currency. We mention here KARMA [Vishnumurthy et al., 2003] where each participant of the system has an associated single value named *karma*. A set of privileged nodes named *bank-set* keep the record of *karma*s. The *karma* of one node is increased when resources are contributed and decreased when resources are consumed. A transaction is not allowed to proceed if the consumer has less *karma* than it takes to make the payment for the resource involved. Thus, *karma* is similar with the owned money and regulates the transactions happening on the network. In Tycoon [Lai et al., 2005], resource-allocation is done using an economic principle. Each host runs an auctioneer and consumers actively bid for resources, committing credit for them. In both approaches mentioned above, only one currency is established for the whole network. i-WAT [Saito, 2006] proposes the model of multiple currencies regulating the exchanges happening in P2P systems with various sorts of services delivered. In such networks, a peer can exchange capability that it does not immediately need for an alternative service. Thus, it enhanced a better resource sharing and utilization. We contributed the literature in this respect, by proposing that Service Level Agreements - described in section 1.2.2 to represent the exchange token [Petri et al., 2011]. Jain et al. [2008] present a critical survey about micropayment schemes in P2P systems.

Peers interaction and service exchange can be modeled with the help of game theoretical concepts. In the classical Prisoners' Dilemma game, mutual cooper-

[1] which accounts for more than 80% of total number of peers

ation is the dominant behavior on the long term [Fudenberg and Tirole, 1991]. Thus, it make sense that the tit-for-tat strategy to work well and various long-term established communities to be able to resist based on the principle of the direct reciprocity. Although, because of asymmetry on peer interests and of capabilities, and due to the fact that P2P communities are in general very large with huge populations of peers that interact only infrequently, cooperation based on direct reciprocity might be difficult to achieve.

In P2P networks, indirect reciprocity works also well. We mention here the notion of reputation, in-depth analyzed in section 1.3. Reputation is earned by peers after cooperation in bilateral transactions, behavior reported by third parties. In subsection 1.3.2.1 we performed a detailed review of reputation models, including how this concept is used in conjunction with P2P systems. Reputation represents one of our major areas of scientific contribution described in this thesis.

All these economic models are devised for the long lasting of the P2P community. But, in order to prove the effectiveness of such a model, one should show how the model resists to various attacks, described below.

Free riding, as defined above represents the main threat to P2P systems. In general, self-interested peers want to consume services from P2P systems without contributing. Other attacks and exploits are also of interest.

First, in various economic-based schemes, peers are required to contribute in order to gain benefits from the networks. Peers can collude in order to claim that they received / contributed services from / to other colluders. Referring to the Prisoner's Dilemma mentioned above, many forms of collusion are possible, including (i) false praise - falsely claiming defectors have cooperated or (ii) false accusation - falsely claiming cooperators have defected. Collusion represents a strong attack against reputation systems, because nodes from the group supply false positive opinions between themselves. In general, collusion is favored by the Sybil attack [Douceur, 2002] where one owner can join the P2P system under multiple identities. One can easily develop a Sybil attack in web-based P2P systems where nodes are created based on email addresses. As indicated in subsection 1.3.2.1, several reputation models including EigenTrust [Kamvar et al., 2003] are collusion resistant. As shown in section 1.4, on P2P equipped desktop grids, we

were the first to tackle the peer collusion problem and propose a practical solution. The algorithm tackling collusion in P2P desktop grids represent one of the major scientific contributions of this thesis.

Whitewashing [Feldman et al., 2004] represents another powerful attack in P2P system. After defecting, a whitewasher leaves the system and comes back with a new identity. This attack is very effective against P2P systems incorporating reputation models. In general, reputation is built based on the history of the peer and assumes persistent identities. Thus, such a model can not distinguish between a legitimate newcomer and a whitewasher. Whitewashing raises the question about how to interact with a newcomer? If always cooperating, this behavior will be fully exploited by the whitewashers. If always defecting, this would protect against whitewashers, but will introduce a barrier on the entry on the network [Friedman and Resnick, 2001]. Our latest contribution [Araujo et al., 2011] proposes a method against collusion that also prevents whitewashing.

In designing P2P systems and the interaction protocols between peers, researchers should contribute with mechanisms that leads towards the stability of the system and increased welfare in the society, in the conditions of the self-interested behaviors of the peers. Such mechanisms should induce truthful revelation of the peers' profiles and truthful biding in resource allocation mechanisms. We have contributed in this direction [Silaghi et al., 2011] by supplying a framework to construct time-constrained SLA strategies, enabling peers to learn good strategies for them, while maintaining a fair resource distribution at the global societal level.

1.2.2 Service Level Agreements and Service-based P2P Systems

Initially, P2P systems were created for the distribution of files over the Internet. During last years, P2P systems evolved towards Service-oriented systems in which nodes interact for various service deliveries. The concept of Service Level Agreement (SLA) regulates the service delivery in service-oriented architectures (SOA). During this P2P thesis, SLAs represent a central concept, being the ob-

ject of the peers' interaction. In this subsection, we will conceptually introduce the service level agreements and the domain of service P2P systems.

Service Oriented Computing is a paradigm that utilizes services as central elements enhancing the development of distributed systems in heterogeneous environments [Georgakopoulos and Papazoglou, 2009]. Within this context, a Service-oriented architecture (SOA) is a collection of services that communicate with each other and cooperate for creating the applications addressing various needs of organizations. In a SOA, services are technologically independent and they encapsulate an application being delivered as a service for others needs. As indicated in Barry [2003], a SOA is characterized by the following principles:

- logical view: the system consists of various programs, databases and business processes
- message orientation: each service consumption involves a message exchange between at least two entities: the service provider and the consumer
- granularity: one service can define a number of operations and complex messages at a certain granularity
- network orientation: services are exchanged over a network, e.g. in a P2P system
- platform neutral: services are described in platform neutral standards like XML and a service can be consumed in a platform of different sort than the one of the service provider

A service identifies a special resource representing the capability of performing tasks with a coherent functionality. When the service delivery happens on the Internet, we speak about Web services. According to Birman [2005], a web service works as a software system that can be accessed by the users over the Internet. According to W3C consortium [Austin et al., 2004], a web service is defined "*as a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner*

prescribed by its definition, using XML based messages conveyed by Internet protocols".

A service has the following properties [Rosen et al., 2008]: service interface, interface documents, service policies, quality of service (QoS), performance. In our scientific contributions, we will be focusing especially on quality of service property. The QoS is described within the service level agreement statement (SLA), which represents a contract between the service provider and the consumer and defines the level of the service properties to be delivered. Gridipedia supplies a good description for the service level agreement concept, description adopted by the FP7 S-Cube Network of Excellence[1]: "*Service Level Agreement is a formal written agreement made between two parties: the service provider and the service recipient. The SLA itself defines the basis of understanding between the two parties for delivery of the service itself. The document can be quite complex, and sometimes underpins a formal contract. The contents will vary according to the nature of the service itself, but usually includes a number of core elements, or clauses. Generally, an SLA should contain clauses that define a specified level of service, support options, incentive awards for service levels exceeded and/or penalty provisions for services not provided. Before having such agreements with customers the IT services need to have a good quality of these services, Quality management will try to improve the QoS, whereas the SLAs will try to keep the quality and guarantee the quality to the customer*". We formally approached SLA-regulated open environments by defining a reputation model for SOA architectures [Silaghi et al., 2007b], fully described in section 1.3. The formalization of the SLAs in terms of economic theory is found on subsections 1.3.4 and 1.5.2.1.

In Web-based environments, OGF defined the WS-Agreement protocol [Andrieux et al., 2007] as a language to describe the agreed service usage details. The SLA description includes the Service Level Objectives, which in fact represent the QoS properties. In environments based on WS-Agreement described contracts, SLO negotiation is envisaged as a bilateral offer-counteroffer message exchanges; thus with strong grounds on the economic theory of service exchange. WS-Agreement Negotiation protocol [Waeldrich et al., 2011] describes in detail

[1]http://www.s-cube-network.eu/km/terms/s/service-level-agreement# consulted on 23 April 2012

how this conversation between a provider and a consumer should happen. We contributed the scientific literature in this domain by proposing a particular negotiation behavior [Silaghi et al., 2011], given that time constraints restrict the negotiating agents. The SLA negotiation setup is described in subsection 1.5.2.1 and subsection 1.5.2.2 details about scientific contributions regarding SLA negotiation in computational grids.

Besides WS-Agreement widely used in the Web environment, other formalizations exist for SLAs, including SLAng [Lamanna et al., 2003], Rule-based Service Level Agreement [Paschke, 2005] and Web Service Level Agreement [Ludwig et al., 2003].

In the last years, P2P systems went away from its simple file distribution role. Nowadays, various services are delivered with the help of P2P systems, including computation power, messaging, VOIP, video streaming and others. QoS plays a fundamental role in such systems and, in general, SLAs regulate the peers interactions. Among the first service-based P2P systems, we mention the desktop grid platforms like Seti@Home, described in more detail in subsection 1.2.3, the I-Way system [DeFanti et al., 1996] which enables researchers to use multiple interconnected supercomputers and advanced visualization systems to conduct large scale computations, the FIPER environment [Sobolewski, 2002] which describes a network of interconnected services hiding legacy applications and programmed with object-oriented Java abstractions. Goel et al. [2007] describe a service based P2P overlay allowing mapping a federated services into business processes. Various software tools allow easy programming and deployment of systems based on bilateral service-based interaction. We mention here toolkits like JXTA [Gong, 2001], Jini [Newmarch, 2006], or the widely used web service concept.

Regarding our concern, P2P service systems represent the proper environment for allowing open and wide participation, representing the fertile soil for planting economic theory principles in the design of distributed computer systems. As indicated in Krishnan et al. [2006], the main economics issues of interest in service-based P2P networks represent the incentives for participation, user behavior and motivation, reputation and trust and the intellectual property. Incentives are strongly related with the free-riding behavior analyzed previously. Either

pricing resources, or using a taxation scheme or micropayments could represent a solution. Or, when pricing is not effective because of peers anonymity, quality of service regarding the delivery might represent an alternative. User behavior represents a well-studied research question in such networks. In general, users are self-interested, and network mechanism design should induce a community accepted behavior and exclude out of the network the saboteurs. Sabotages can happen, mostly within the standard behaviors explained in the previous subsection, but, for every network protocol, a full game-theoretical analysis is a valuable contribution. As we will see in section 1.3 trust and reputation is one of our major concern.

Service based P2P systems are found as effective alternatives for increasing the satisfaction of large consumer communities. Li and Lee [2010] show that service quality of peer-produced services increases with the heterogeneity of the participants and with their number. Thus, it is worth studying issues concerning P2P systems, as peer-generated content is a valuable asset on nowadays computing.

1.2.3 Desktop Grids

This subsection will introduce the topic of desktop grids. Desktop grids deployed over Internet represent a space where anonymity and lack of control can induce malicious behaviors. Our scientific contribution in desktop grids advances the state-of-the-art regarding sabotage [Domingues et al., 2007]. In relation with the rest of the thesis, desktop grids can be considered as P2P systems, where peers contribute individually with computing power. Further, P2P technologies might be employed in desktop grids to leverage some drawbacks, like the cost of data distribution. In this respect, our results were among the first to link the problem of sabotage with the problems induced by the usage of P2P concepts. In this subsection, we will shortly describe the basic concerns in desktop grids and volunteer computing, with a emphasis on P2P-related aspects.

The last decade we faced increased computing requirements by complex application, resulting a huge development of high performance computing. HPC is primarily supported by classical grids systems [Foster et al., 2001], in general hosted by universities and research centers. Grids require huge spendings in

computer hardware and, to manage and use such systems, one needs specific IT support and good knowledge of computing. Desktop grids emerged as a cheap alternative to classical grid systems, by harnessing the idle computing power of home desktop computers. The Condor project[1] [Thain et al., 2005] developed at University of Wisconsin-Madison represents the first system to aggregate collections of distributively owned computing resources. Bayanihan computing proposed by Sarmenta and Hirano [1999], BOINC [Anderson, 2004], Entropia [Chien et al., 2003], XtremWeb [Cappello et al., 2005], GridSAT [Chrabakh and Wolski, 2006] or Ourgrid [Mowbray, 2007] represents other systems organized on the same principles: aggregating desktop computing power from various geographically distributed locations, including Internet. We need to make a first distinction in what regards desktop grids. If the computing power is voluntarily donated in the system by (anonymous) Internet users, we speak about *volunteer computing*. If the desktop grid platform is installed on the computers from an organization and the origin of aggregated resources is controlled, we speak about *enterprise desktop grids*.

Similar with grid systems, the main usage of desktop grids is in scientific computing. In general, desktop grids were preferred by researchers that were not owning a budget to afford a grid. Nowadays, we notice the spread of grids and desktop grids to supply the computational power required by business applications and even the integration between desktop grids and grids or clouds. We mention here (i) the FP7 EDGeS project[2] that created an integrated grid infrastructure by seamlessly integrating various desktop grid solutions (like BOINC and XtremWeb) with EGEE type of service grids and (ii) Aneka[3] that allows provisioning of private cloud resources including desktop, cluster and virtual data centers and public clouds to various computing needs.

Bag-of-tasks represents the basic computing model adopted by desktop grids. The computing project should be split in multiple independent tasks that can be parallel deployed on the desktop resources. In general, those tasks should not

[1] http://research.cs.wisc.edu/condor/ consulted on 25 April 2012
[2] http://edges-grid.eu/ consulted on 25 April 2012
[3] http://www.manjrasoft.com/products.html

communicate between them. Thus, the architecture of a desktop grid is *master-worker*, with a master owning the bag-of-tasks and distributing the tasks to the workers installed on the desktops. The scheduling process can be of two sorts: (i) *push*: where the master initiate the scheduling process and (ii) *pull*: where the ready workers initiate the scheduling process by asking the master for new tasks. Nowadays, desktop grids become ready for novel computing paradigms, including Map-Reduce [Marozzo et al., 2011; Tang et al., 2010; White, 2009].

The classical bag-of-tasks model assumes that each task contain its data and tasks do not communicate between them. However, in scientific computing, parameter-sweep applications are very popular. In fact, such applications represent experiments on a set of data, run with varying input parameters. If the dataset is big, distributing it to the workers might represent a problem, if the bag-of-tasks model is employed as-it-is. The data distribution problem is similar with the one that favorized the popularity spread of BitTorrent, which leveraged the bandwidth costs from the seeders to the shoulders of the downloaders. In desktop grids running parameter sweep applications, would make no sense to deliver the same (in general big) amount of data with every task scheduling. Thus, solutions for clever data distribution and wise scheduling are requested. Using BitTorrent represents an affordable alternative for data distribution Fedak et al. [2008], and this solution represented a first step towards inter-joining the desktop grids with classical P2P systems. Nowadays, a lot of desktop grids projects are organized using P2P principles, in various areas, including architectural design, user management, resource management - including identification, monitoring, discovery and utilization, and security and reliability management. A full taxonomy of peer-to-peer desktop grids was developed by Zhao et al. [2011].

Below, we mention several approaches employing P2P concepts in desktop grids. Cohesion [Schulz et al., 2008] uses a P2P concepts and a flexible architecture in order to deal with the high volatility of desktop grid nodes. It is programmed in JXTA [Gong, 2001] and employs an unstructured P2P model for flexible task migration and load balancing. CCOF [Zhou and Lo, 2004] organizes nodes in a CAN-based DHT according with their time zone. CCOF scheduler aims to benefit from the lower workload of the desktop grid nodes during the night time in order to handle deadlines for the bag-of-tasks projects. Ourgrid

[Mowbray, 2007] groups users in communities. As everyone can join the system to contribute and benefit, in order to prevent free riding, Ourgrid uses a reciprocation method described in Andrade et al. [2007]. PeerStripe [Miller et al., 2007] uses P2P routing to devise a self-organizing storage system for managing large data files around nodes in a desktop grid. Mainly, it addresses the same data distribution problem like Bitdew [Fedak et al., 2008]. Another solution to data distributed was given part of the FP7 EGDeS project by Kelley and Taylor [2008], proposing the ADICS framework. Files are stored and replicated on peers and a simplified Peer-to-Peer middleware manages the underlying unstructured overlay.

In our scientific contribution we approached volunteer computing desktop grids. While anonymous Internet users contribute voluntarily to such desktop grids, it appears the problem of sabotage. Initially, sabotage was approached within the master-worker model under the bag-of-tasks abstractions, considering that worker nodes do not communicate between them. Our scientific contribution was among the first to consider that nodes can communicate between them and exhibit coordinated behaviors, like results collusion. We postpone the discussion about the sabotage problem in desktop grids up to section 1.4, where we present our results and in subsection 1.4.2 where we introduce related work in this respect.

1.3 Reputation Management

In this section we present the reputation model we designed for dealing with autonomous entities in collaborative computing. Initially, the model was devised for service oriented computing [Silaghi et al., 2007b]. Next, the model was incorporated in a larger mechanism for virtual organization formation, allowing collaboration between entities in virtual breeding environments [Arenas et al., 2010]. Reputation helps in providing a mechanism for service provider selection and usage control.

In collaborative systems, a set of organizations shares their computing resources, such as compute cycles, storage space, or on-line services, in order to establish Virtual Organizations aimed at achieving common tasks. The formation and operation of Virtual Organizations involve establishing trust among their members and reputation is one measure by which such trust can be quantified and reasoned about. We contribute to research in the area of trust for collaborative computing systems by presenting a model for reputation management for Grid Virtual Organizations that is based on utility computing and that can be used to rate users according to their resource usage and resources and their providers according to the quality of service they deliver. We also demonstrate, through Grid simulations, how the model can be used in improving completion and welfare in Virtual Organizations.

1.3.1 Research objective

In collaborative systems, a Virtual Organization (VO) can be defined as a set of users and real organizations that provide resources, such as compute cycles, storage space, or on-line services, for users to exploit for a common goal. Examples of common goals include large-scale distributed computing research projects [Britton et al., 2009] or inter-organizational business applications such as Grid-based supply chains [Blasi et al., 2008], among others. In such collaborative systems, trust management is a fundamental problem as resource owners must share their resources with unknown organizations as well as ensuring that all users abide by the VO agreement to which the resources have been allocated.

This research investigates how to exploit reputation systems for managing Grid-based VOs. Reputation is one measure by which trust among different members of a VO can be quantified and reasoned about. We focus on Grids where the availability of resources and user tasks is highly dynamic, and both resource providers and users have to compete for providing and employing resources. Reputation systems are then used to manage reputation of resource providers, according to the Quality of Service (QoS) provided, as well as reputation of users, according to their usage of resources.

Our reputation model is grounded on a utility based reputation model for service computing. A reputation system gathers, aggregates and distributes feedbacks about participants behaviour. The feedback is usually provided as an a posteriori operation requiring human intervention. This way of building reputation is useful in semi-automated contexts, such as electronic marketplaces where users rate sellers, but it becomes a limitation in fully automated contexts such as Grid-based collaborative systems [Foster et al., 2001]. In some Grid applications, the resources allocated to a users job are unknown to the user, and such allocation could change during the job execution, making it difficult to obtain users feedback. The model introduced in Silaghi et al. [2007b] and presented in this chapter overcomes that limitation by representing users feedback as utility functions, which takes as input the information provided by trustworthy monitors after a transaction. The utility functions reflect the satisfaction a user perceives after consuming a service; this information is thus used to create reputation about particular resource providers and users.

The main contribution is to study the impact of applying reputation in Grid-based VOs. Resource-providers reputation can be used by resource brokers in order to improve allocation of user tasks by selecting reputable providers. Conversely, users reputation can be used by resource providers in order to define security level for users; low-reputable users would be assigned tight measures when accessing a resource.

The structure of our presentation is the following. Section 1.3.3 describes our model of VOs. Then, Section 1.3.4 introduces our reputation model, a utility-based model that uses information provided by monitors to rate entities within a Grid. Next, Section 1.3.5 presents the system architecture and gives an example

of a usage scenario. Then, Section 1.3.6 shows experimental results and discusses the use of reputation for both brokering and controlling resource usage. Finally, section 1.3.7 concludes the scientific contribution and highlights future work.

1.3.2 Reputation-based trust management systems

This section reviews the main reputation-based trust systems. We directed our analysis trying to identify how these systems fulfill the requirements of computational grids, anticipating a further inclusion of reputation-based technologies for bringing trust in computational grid systems. We analyze a wide area of developments, from reputation systems used in e-commerce to systems coming from agent research, P2P and Grids.

Trust and reputation systems have been recognized as playing an important role in decision making in the Internet world [Grandison and Sloman, 2000; Jøsang et al., 2007]. Customers and sellers must trust themselves and the services they are offered. Regarding the grid systems, the fundamental idea is that of resource sharing [Foster et al., 2001]. The grid research was initiated as a way of supporting scientific collaboration, and grid systems were mainly used in e-science projects. Entities from trusted institutions are put together to collaborate and form the grid. However, when grid systems are intended to be used for business purposes, it is necessary to share resources between unknown, un-trusted parties. If one intends to generalize the wide usage of grid systems, the problem of un-trusted parties should be considered. The grid definition of CoreGrid emphasizes the dynamic property of almost every issue: *a fully distributed dynamically reconfigurable, scalable and autonomous infrastructure to provide location independent, pervasive, reliable, secure and efficient access to a coordinated set of services encapsulating and virtualizing resources in order to generate knowledge.* As the CoreGrid survey material on trust and security acknowledges, modeling trust is of great importance for the future developments of the grid [CoreGrid, 2005].

Reputation-based trust systems were mainly used in electronic markets, as a way of assessing the participants. In a lot of such environments, they proved effective as the number of participants was large and the system was running a

sufficient amount of time [Resnick et al., 2000]. But, there are still a lot of issues under study as not everywhere reputation systems were fully effective.

In grid systems, usually trust is constructed and maintained through security mechanisms [CoreGrid, 2005]. Technical advances go toward enabling one point sign-on for an entity in the system, considering that the entities belong to some generally trusted organizations. But, as the scope of grid enlarges to ubiquitous and pervasive computing, there will be a need to assess and maintain the reputation of entities, once they are allowed to participate in the grid. Our work intends to evaluate the suitability of existing reputation management systems with regard to the grid security requirements.

Several reviews addressed the problem of trust and reputation models for various domains. Grandison and Sloman [2000] survey several existing trust models in 2000, focusing on Internet applications. The main contribution merit of this paper is a good conceptual definition for trust and the establishing of some trust properties. They do not address computational trust management models, while they focus more on trust gained by certification. Reputation is not addressed in this review.

Regarding trust in E-Commerce applications, Manchala [2000] evaluates some trust metrics but they do not address the reputation problem. Before presenting their developments for trust management through reputation, Zacharia and Maes [2000] review some systems live in 2000 that address reputation management in e-commerce sites. Regarding on-line trading environments, Dellarocas [2005] analyzes reputation mechanisms from a game-theoretical point of view. He allows opportunistic players to take part of the game and his analysis is fully based on mathematics developments.

Suryanarayana et al. [2004] address the topic of peer-to-peer applications, analyzing properties of reputation systems related with peer-to-peer requirements.

Jøsang et al. [2007] refer to the problem of on-line service provision, but they address the topic of trust and reputation systems from a general point of view, covering applications from both e-commerce and p2p. They analyze the computational engines classified according with their category: simple summation, Bayesian systems, discrete trust models, belief models, fuzzy models and flow

models. Also, they describe some reputation systems live at the time moment of the paper. They do not make clear which system to which category belongs to.

Sabater and Sierra [2005] review some works regarding reputation as a method for creating trust from the agent-related perspective. They do not categorize the described models, but they try to find how those models are related with some theoretical requirement properties for reputation. The review of Ramchurn et al. [2004a] considers also a multi-agent perspective while debating on the notion of trust.

In section 1.3.2.1 we will develop the concepts of trust and reputation, establishing some desirable properties a reputation system should fulfill. These will be the properties we will look for when analyzing a reputation system. These properties were extracted in close relationship with the requirements that grid imposes, on the movement toward a widely used grid infrastructure. Section 1.3.2.2 described the main advances in reputation research. We collect studies from a wide area of computer science: multi-agent research, knowledge engineering, grid systems, learning, information retrieval, e-commerce, etc. Section 1.3.2.3 will shortly point on the usage of reputation systems for enhancing grids with fault-tolerance and to improve resource management. Section 1.3.2.4 will conclude this review of reputation models.

1.3.2.1 Trust and reputation

This subsection defines the concepts of trust and reputation and identifies the main properties that trust and reputation management should fulfill, considering also the requirements imposed by the computational grid systems.

Trust

According to Gambetta [1988], trust refers to the subjective probability by which an individual A expects that another individual B performs a given action on which its welfare depends. This definition taken from sociology is very popular in computer science today.

From the business point of view, the European Commission Joint Research Centre defines trust as *the property of a business relationship, such that reliance*

can be placed on the business partners and the business transactions developed with them [Jones and Morris, 1999].

Marsh [1994] was one of the first to define the trust concept from a computational point of view. He takes the definition of Deutch [1962] which states that trusting behavior occurs when an individual perceives an ambiguous path, the result of which could be good or bad, and the occurrence of the result is dependent on the actions of another person, the bad result being more harming than the good result beneficial. If the individual chooses to go down that path, he can be said to have made a trustful choice. Marsh agrees that trust implies some degree of uncertainty and hopefulness or optimism regarding an outcome, being subjective and dependent on the views of the individual.

A recent definition of trust is the one of Grandison and Sloman [2000]: *the firm belief in the competence of an entity to act dependably, securely and reliably within a specified context.*

Jøsang et al. [2007] makes a difference between *reliability trust* as a subjective probability, defined according with Gambetta [1988] and the *decision trust* as being the extent in which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible.

Falcone and Castelfranci [2001] presents a cognitive view about trust, which is applied in the context of task delegation. When delegating a task, an agent a might evaluate the trust it places in another agent b, considering the different beliefs it has about b: (1) the competence belief: b is competent to do the task, (2) the disposition belief: b actually will do what a needs, (3) the dependence belief: a believes at least that it is better to rely on b for the task than not to rely on it, (4) the fulfillment belief: a beliefs that the task can be done, (5) the willingness belief: b intends to do what it has been proposed to do, (6) persistence belief: b is stable enough in this intentions, (7) the self-confidence belief: a should belief that b knows it can do the task and (8) motivation belief: b has some motive to help a.

The above cognitive approach is worth for consideration in grid systems as a theoretical foundation for empowering grids with trust management, considering the task delegation and resource management requirements of grids. But,

when implementing trust management mechanisms, a lot of studies employed the subjective probabilistic view, as being more suited to a computational approach.

Reputation

Reputation is what is generally said or believed about a person's or thing's character or standing [Jøsang et al., 2007]. They argue that reputation is a mean of building trust, as one can trust another based on a good reputation. Therefore, reputation is a measure of trustworthiness, in the sense of reliability.

According to Abdul-Rahman and Hailes [2000], a reputation is an expectation about and agent behaviour based on information about or observations of its past behaviour.

This last definition emphasizes the two main sources for building the reputation of an entity: the past experience and the collected referral information. Yu and Singh [2002] go further and identify the challenges a reputation management system should address: (1) how an agent rates the correspondent based on the past interaction history, (2) how an agent finds the right witnesses in order to select the referral agents and (3) how the agent systematically incorporates the testimonies of those witnesses.

Other authors argue that reputation is solely gathered from the social network in which the agent is embedded [Sabater and Sierra, 2005]. Therefore, trust can be built from (1) the confidence an agent derives from past interaction and (2) the reputation the agent acquires from the social network [Ramchurn et al., 2004a]. The first source of trust is named *direct trust* while reputation represents an *indirect trust* source.

We will allow reputation to be assessed both from past experience and referrals. Therefore, reputation-based trust systems can be classified in 2 main categories: systems that use only direct trust measures and systems that use both direct and indirect trust.

Properties of reputation-based trust models

Regarding grid systems, the CoreGrid survey material on Trust and Security [CoreGrid, 2005] acknowledges about the importance of trust management in grids and presents how trust is bring in using security issues. Up to date, reputation based models are barely considered for classical grid systems. As the long

term future of the grid is to provide dynamic aggregation of resources, provided as services between businesses, new architectures and detailed mechanisms for bringing together arbitrary resources are required. These architectures should federate security and trust, as ones of the most significant issues [Ahsant et al., 2006]. On the basis of the OGSA architecture [Foster et al., 2005],

- WS-Trust defines a protocol by which web services in different trust domains can exchange security tokens for use in the WS-Security header of a SOAP message.
- WS-Federation describes how to use WS-Trust, WS-Security and WS-Policy together to provide a federation between security domains.

Therefore, in classical grids, trust is achieved through security mechanisms. Attempts like the ones of Jurca and Faltings [2005b]; Kerschbaum et al. [2006]; von Laszewski et al. [2005] are among the few to use reputation tools for managing virtual organizations as in grids. Other approaches ([Buchegger and Boudec, 2005; Rebahi et al., 2005]) tackle mobile ad-hoc networks. But, the most ones ([Despotovic and Aberer, 2005; Gupta et al., 2003; Jurca and Faltings, 2005a; Kamvar et al., 2003; Singh and Liu, 2003; Xiong and Liu, 2004; Zhao et al., 2005]) address resource management as in P2P applications.

These attempts are mainly based on the developments and requirements identified in P2P systems, as P2P are the models most closed to the fully dynamic and distributed resource management requirements envisioned by the future grids.

Several properties are common to most reputation-based trust models, without any regard of their applicability:

- The computational model: Because grids are based on the distributed computational model, the first property of interest is if the trust mechanism is centralized or decentralized. Centralized models have the disadvantage of a single-failure point, therefore, regarding desktop grids, decentralized systems would be preferable. In classical grids, where security is achieved through certificates and central certification authorities exist, a centralized

model could also be of interest. In such systems, one can think to a reputation service in order to be interrogated about the reputation of a user or, more generally, of a resource. This reputation service in this case is a point of centralization.

- Metrics for trust and reputation: when referring to a metric for trust and reputation we consider the value that express the reputation (and trust) of an entity as provided by the reputation mechanism. We must make a distinction between the reputation value of an agent and the feedback one is required to provide at the end of a transaction. Continuous metrics are considered more expressive than discrete ones. Usual, these values are scaled between -1 and 1, or between 0 and 1. If the reputation scheme uses values scaled between 0 and 1 these values can have the meaning of a probability.

- Type of reputation feedback: reputation information might be positive or negative one. Some systems are based on collecting both type of information with regard to an entity, while other systems are based only on negative / positive information. Regarding an accomplished transaction, the reviewer can supply with binary, discrete or continuous values. Again, continuous values are more expressive but for the sake of simplicity, a lot of approaches use discrete feedback and later on aggregates this feedback in continuous reputation or trust.

- Reliability: the trust model should help the users to defend themselves against malicious information, including trust values propagated by other users into the system. The system is reliable if this property is accomplished. Almost all researchers reported that their reputation-based system is reliable for their specific problem under study.

With regard to P2P applications, the following properties might be of interest [Suryanarayana et al., 2004]:

- Local control: in decentralized applications, data are stored at various nodes in the system. As global trust might be stored at the entities in the system,

is important not to allow those entities to change the trust and reputation values they maintain. Local control property will have the value yes for those models accomplishing this property.

- Bandwidth cost: in P2P applications, bandwidth is of great importance, as peers communicate via message transfer. In a reputation-based trust system, peers might exchange reputation information, which can increase the bandwidth cost of the network. We desire to have the lower possible bandwidth cost. When a referral network is used to acquire reputation information and if a P2P distributed approach is considered for data storage, the bandwidth cost is increased.

- Storage cost: in a decentralized architecture, the nodes on the grid store trust information about other nodes. One would desire to have as few as possible data replication, and therefore, the storage at each node for trust information should be as less as possible. In a centralized setup, usually, the trust data is stored in the central node and storage cost is less important at the node level. We should acknowledge that the storage cost increases linearly with the number of entities in the system.

- Scalability: the trust model should scale with the number of nodes. Bandwidth and storage costs also increase with new nodes added to the grid, but the trust model should be built in such a way to scale well. We reported the scalability property according with the size of the experiments the authors performed in their papers.

With regard to grid systems we consider the following two properties as of particular importance:

- SLA or QoS negotiation: some reputation models are directly applied for negotiation of service level agreement (SLA) or quality of service (QoS) between 2 parties like a service consumer and producer. In most of the cases, the items to be negotiated and how each party fulfilled the agreements on the specific items part of a SLA are directly incorporated in the direct trust component.

- Trust aggregation: we refer to trust aggregation if the model allows to aggregate trust on an organizational basis. This property is of great importance in the context of VO formation and operation as allows one (1) to obtain the trust and reputation for a VO based on the individual trust on its members or (2) to infer the trust or reputation for an individual based on the trust and reputation of organizations the individual belongs to.

Table 1.1 depicts a summary of how the models we detailed on section 4 accomplish with these properties. Where information about a specific property was not found on the underlying research, we used the *na* (not available) notation.

Table 1.1: Summary of comparison between reputation-based trust systems

	Centralized	Metric for trust and reputation	Type of feedback	SLA or QoS negotiation	Trust aggregation	Local control	Bandwidth cost	Storage cost	Scalability
eBay	yes	discrete, infinite	discrete	na	no	na	na	na	5
Amazon	yes	[1,5]	discrete	na	no	na	na	na	5
Abdul-Rahman & Hailes [1]	no	discrete	discrete	na	no	no	5	5	na
Singh et al. [48]	no	VSM	na	QoS verification	no	no	5	3	3
Despotovic & Aberer [9]	no	[0,1]	na	QoS verification	no	yes	5	5	3
Josang' subjective logic [25]	no	[0,1]	binary	na	no	no	5	3	na
Yu & Singh [53]	no	[0,1]	continuous	na	no	no	5	3	3
SPORAS and HISTOS [54]	yes	[0,3000]	continuous	na	no	na	na	na	3
REGRET [42]	no	[-1,1]	continuous	QoS verification	yes	no	3	3	5
Ramchurn et al. [36]	no	[0,1]	discrete	SLA-oriented	yes	no	3	3	na
FIRE [22]	no	[-1,1]	continuous	QoS verification	no	no	4	3	3
Gupta et al. [21]	semi	discrete, infinite	na	QoS verification	no	na	3	1	5
TrustMe [47]	no	na	na	na	no	yes	4	1	4
EigenTrust [31]	no	[0,1]	binary	na	no	yes	5	5	3
PeerTrust [52]	no	[0,1]	continuous	na	no	yes	5	5	3
P-Grid [2]	no	na	negative information	na	no	yes	5	5	4
NICE [46]	no	[0,1]	continuous	na	no	yes	3	3	4
GridEigenTrust [50]	yes	[0,1]	continuous	QoS verification	yes	no	1	na	na
PathTrust [32]	yes	(0,1]	binary	na	no	na	na	na	4
Jurca & Faltings [27]	semi	[0,1]	binary	QoS verification	no	na	3	3	4

1.3.2.2 Reputation systems

In this section we will describe the main reputation systems built up-to-date in the research community. We will start our discourse with a short introduction in the game-theoretical foundations for reputation models.

1. The game-theoretical approach for reputation

From the theoretical point of view, economics approaches the problem of reputation in a game-theoretical framework. Agents (players) are continuously playing the same game. When an agent plays the game repeatedly in the same way, it is assumed that the player builds a reputation for playing certain kinds of actions and the rest of the players will learn this reputation. The main concern is when and whether a long-lived player can take advantage of a small probability of a certain type or reputation to effectively commit him to playing as if he were that type [Fudenberg and Tirole, 1991]. A related question is if reputation models will help one to pick and choose among the many equilibriums of an infinitely repeated game.

Considering long-lived players playing repeatedly the classical prisoner‘s dilemma game, allowing for incomplete information about players type and allowing for building a reputation, the theory can prove that the long-term outcome of the game will be “to cooperate” for both players, although the sole short and long term Nash equilibrium is to defect [Kreps and Wilson, 1982]. Also, in games with only one long-lived player and short-living opponents, considering reputation in the presence of incomplete information lets the theory to prove that the “intuitive” outcome will happen. Fudenberg and Levine [1992] analyzed the chain-store game with this respect. These are good examples to demonstrate how effective reputation can be in gaining a bigger payoff from incomplete information situations where some agents have to consider a decision making.

Game theory usual helps one to demonstrate that is worth to consider reputation information when analyzing the outcome of some competing situations with incomplete information. It is also worth to notice that game theory usual considers reputation as being built only from previous experience of the player within a specific context.

Such approaches were considered for analyzing sensitive options a reputation

system designer might have. E.g. Dellarocas [2006] proved that a reputation system should not update the reputation of players immediately after a transaction finishes. Rather, if the players‘ reputation is updated with a-priori established time frequency, the players can learn the reputation of opponents in the game and more cooperation can be induced.

2. Reputation in Internet sites

Internet sites mainly use sumation-based reputation systems. These systems are based on counting all votes or grades an entity receives. The votes can be simply counted on the behalf of the user or they can be averaged or weighted. As summation-based reputation systems are mainly used in e-commerce marketplaces, they are mostly centralized. Their big advantage is the simplicity of the reputation scheme. This makes the reputation value to be easily understood by the participants and allows a direct conversion between reputation assessment and trust. The most widely known reputation system of this kind is eBay. Other systems are Amazon, Epinios, BizRate etc.

eBay

The most simplistic approach for assessing reputation is the summation scheme of eBay. eBay[1] is an auction-based e-commerce site for sellers and buyers with millions on items to bid for. The reputation management system is a transaction based one. After the end of an auction, the buyer and the seller have the opportunity to rate each other‘s performance with either 1 (positive), 0 (neutral) and -1 (negative). The reputation of a user is the sum on these individual feedback and it is a common knowledge into the system. The system stores and manages the reputation centrally. New users receive no reputation and a user may leave the system and rejoin with another identity. The advantage of this reputation scheme is that the reputation measure is easily understood by the participants and therefore, the reputation information can be quickly transformed in a trust knowledge.

In eBay, most of the feedback is positive. Sellers receive negative feedback only 1% of times and buyers 2% [Resnick and Zeckhauser, 2002]. Therefore, the negative information is the most valuable one in the reputation database.

[1] http://www.ebay.com

Jøsang et al. [2007] classifies this reputation system as primitive, but, as Resnick et al. [2006] proves, this primitive reputation system is validated by its long time existence, acknowledging the Yhprums Law: systems that shouldnt work, sometimes do, or at least work fairly well.

Similar feedback summation methods were proposed in other e-commerce websites. Beside summation, averaging the feedback or weighting it was considered.

Amazon

In the Amazon[1] bookstore, reputation is assigned to books and to reviewers. Regarding the books, the reputation of a book is the average score the book received from its reviewers. A reviewer can assign to a book between 1 and 5 stars. Each reviewer has its own reputation. Each time a review is considered helpful by a user, the reviewer receives a vote. The reviewers are ranked based on the votes they received from users.

Similar with eBay, the Amazon reputation system is a centralized one, the reputation is a common knowledge and it has the advantage of simplicity. Anyway, in Amazon, the reputation does not have such a great impact on the whole marketplace, as it can only affect the buying decision, not the price at which the transaction happens. It is also expected that reviewers to receive positive feedback, but, unlike in eBay, Amazon does not display how many negative votes a reviewers received. Amazon reputation system is not a transactional one, as one can vote a reviewer even without buying the item under review. Epinions[2] is another system that offers reviews about products and services in a similar way like Amazon.

This kind of reputation systems is not too valuable for our concern in grids, as they do not allow reputation to directly influence the transaction execution in the system.

3. Reputation models based on referrals network

Building trust can be based not only on the past interactions between entities but, also considering the social networks the entity belongs to and the referrals the entity can obtain using the social network. Singh et al. [2001] defined the

[1] http://www.amazon.com
[2] http://www.epinions.com

concepts of agent communities and social networks. The members of an online community provide services and referrals for services to each other. A participant in a social network has reputation for both *expertise* (providing good services) and *sociability* (providing good referrals). This approach is widely considered in the agent research community.

The referrals network described by Singh et al. [2001] is an agent abstraction for the trust model proposed by Abdul-Rahman and Hailes [2000]. Both approaches propose a reputation learning model that updates the sociability reputation of a user according with the outcome of the interaction with that user.

A lot of research was developed considering this approach. The items under study are the way the reputation is stored in the system, how referral information is aggregated, which learning model is used. This section will develop this sort of referral systems.

Abdul-Rahman and Hailes [2000]

Abdul-Rahman and Hailes [2000] propose a model for computing the trust for an agent in a specific context based on the experience and recommendations. Like with the summation models, trust values are discrete: very trustworthy, trustworthy, untrustworthy and very untrustworthy. Each agent stores the trust values for the agents she interacts with, therefore, the trust model is distributed. Each agent also stores the recommender trust with respect to another agent. The recommender trust value are semantic distances applied for adjusting the recommendation in order to obtain a trust value. They propose a method for evaluating and combining recommendations and updating the trust value. As the model is based on the set theory, each agent has to store all history of past experiences and received recommendations. On a system with a lot of participants and frequent transactions, each agent should have a large storage with this respect. Regarding the network traffic, this is caused by the messages exchanged between agents in order to get reputation information.

The authors provide an example of applying the reputation management scheme, but no computational analysis is provided.

Singh et al. [2001]

Considering the same basic assumptions as Abdul-Rahman and Hailes [2000], Singh et al. [2001] further refine the referral model. Therefore, they assume an agent places queries for services and the responses are of two types: service ex-

pertise and referral. Each query and response is expressed as a vector of required expertise. Responses are evaluated based on the similarity between the received service expertise answers and the received referrals weighted with the trust (sociability) in that agent. According with this representation in the vector space model (VSM), each agent has to store its area of expertise and the models of the peers, including peers‘ expertise and sociability. Each agent updates the peers expertise after verifying (by experience) the QoS provided by that peer. If the QoS is bad, therefore, for the whole chain of agents who referred the peer under discussion the sociability measure is decreased. Periodically, each agent decides which peers are worth to keep in its internal model. Therefore, the storage space at each agent is kept in a reasonable limit.

Authors tested the model in a simulated environment with 20 to 60 agents with expertise in 5 fields. The average number of networks was selected as being 4. The main results are the following: (1) the quality of the social network improves over time, (2) the social network stabilizes at an improved quality, (3) when referrals are given, the quality of the system is high than without referrals and (4) a new agent added to an existing stable network will drift toward the neighbours from which it receives improved quality. Another result reported by the authors regards the existence of some privileged agents in the network with a bigger number of neighbours. If this assumption is fulfilled, the overall quality of the system could be improved.

We can observe that with small number of participants in the network, using reputation mechanisms a gain can be obtained in the quality of service assured. This can have some applicability to classical grids, but the strong requirement is that resource and service selection to be an automated task. More, the system is self-organizing and once achieved maturity, a new member is well accommodated by the system. The privileged agents might be assimilated with the central nodes of a classical grid.

Despotovic and Aberer [2005]

Although the work Despotovic and Aberer [2005] refers to P2P networks, because it fully employ the referral network as being a source for obtaining recommendations, we categorize this paper as belonging to this last category. They use a probabilistic approach for assessing peers‘ trustworthiness in a P2P network. They assume the existence of two interaction contexts: *a direct relationship* where

a destination node performs a task and *recommendations* when the destination node acts as a recommender of other nodes in the network. Rather than considering the standard P2P architecture, the graph of nodes is built by linking peers who have one the above-mentioned relationships. Standard models usual weight a recommendation by the trustworthiness of the recommender. Instead, they model the ability of a peer to make recommendations, which is different from the peer trustworthiness.

Each peer j has associated innate probabilities for performing honest or dishonest with others. Other peers when asked about the performance of j may again lie and misreport. Assuming a probability l_k that a peer p_k lies, one can derive the probability of observing a good or bad report from peer k about peer j. Given a sample of independent reports about peer j, one can compute the likelihood behaviour of j, which in turn, depends on the internal probability of agent j for performing honest. Maximizing this likelihood, one can obtain the probability associated with a peer.

The authors state that good predictions can be obtained with 10-20 reports (direct referrals) retrieved. A peer i can learn the misreporting probability l_k by previous experience with peer k, by asking peer k to report about the service quality that peer produced in bilateral direct interactions. Therefore, a full probabilistic model is obtained for predicting the probability of a peer to be honest.

The setup was simulated in an environment with 128 peers, varying number of random direct interactions (from 20 to 100) and varying percentage of liars (0.1 to 0.5). The mean absolute prediction error is low when the proportion of liars is small. The worse results are obtained when half of the population lies and the number of direct interaction is reduced (20).

The authors entered further details, as considering several services provided by the peers of agents and a normal distribution for each peer with regard to the provided QoS. The average QoS provided by a peer is internal to its model. They analyze the following pattern of behaviour: a service provider j provides a service of quality x to a peer j. If j is honest, then it will report the quality x when requested. On the other hand, j will be liar and will report a quality chosen randomly from a normal distribution. Within this setup, they show that a maximum likelihood estimation method can be used to accurately predict the future performance service providers, given the reports are based on their past

provided qualities.

Testing this second setup in a network with 128 peers, with 10 to 50 interactions per peer, proportion of liars varying from 0.1 to 0.4, 4 services available on the network and the standard deviations of peers performing a service being 0.3, they obtained good misclassification rates regarding the expected service quality.

This approach is worth for consideration in both classical grids and desktop grids. For desktop grids, the approach does not make too much network traffic as only a small number of recommendations are used for each computation. The model each peer stores locally is not too big, being only the misreporting probability each peer learns about its partners. Also, as simulations proved, good predictions can be obtained, increasing the fault tolerance of the system.

Regarding the usage of the model in classical grids, one can predict the misclassification rate for the expected QoS for a service provided by group of peers (which could be a virtual organization), by employing the probabilistic model described above.

4. Belief-oriented trust

These models keep valid the basic assumptions of the referral networks. They refine the above described models by introducing a more sophisticated technique for computing the trust value. The main starting point is that trust is a human belief involving a subject and an object and the trust in a system is a subjective measure. Because of the imperfect knowledge about the reality, one might only have an opinion about trusting an object and this opinion could be a belief, disbelief and uncertainty [Jøsang and Knapskog, 1998]. The roots of this approach are in the Dempster-Shafer theory of evidence. More, this approach is consistent with the theory of Marsh [1994], allowing the existence of two thresholds for expressing trust and untrust beliefs. Trust values are continuous in this case and the storage model can be distributed at the levels of the nodes in the system.

Jøsang and Knapskog [1998]

The Josang's subjective logic [Jøsang and Knapskog, 1998] is a trivalent one, an opinion could have 3 degrees of values: belief (b), disbelief (d) and uncertainty (u), with

$$b + d + u = 1 \qquad \text{with } \{b, d, u\} \in [0, 1]^3$$

Assessing b, d and u from the previous experience of the agent with the object of the trust (which can be another agent or a resource) can be done using the beta distribution function, which is applicable in a space where every event can be successful or unsuccessful.

The subjective logic of Josang introduces the following operators, which can be applied at the internal level of the agent in order to produce the internal trust model.

- the conjunction operator in order to infer a conclusion about a proposition, having two opinions about that proposition
- the consensus operator between independent and dependent opinions
- the recommendation operator, allowing the agent to include in the inference chain the recommendations received from a referral.

The main contribution of Josang‘ subjective logic is a clear representation of the logic each node in the network should posses in order to manage the experience and the received referrals.

Jøsang [1999] shows the compatibility between the subjective logic and the PGP authentication system, demonstrating the usage of the trust values in grid-like networked environments (based on layered certifications).

They applied this reputation mechanism for improving service discovery in a P2P environment in Wishart et al. [2005], combining the reputation computation with distributed hash-table routing structure. In this development, referrals are not used, they pursue building the reputation only based on experience and received feedback.

Yu and Singh [2002]

The model of Yu and Singh [2002] could be more expressive that Josang‘s one as they allow continuous values in order to assess the outcome of a transaction. According with the Marsh approach, Yu and Singh considers two thresholds (low and high) for assessing the belief or disbelief in the trusting relationship. On their model, they directly use the Dempsters rule of combination in order to

aggregate 2 belief functions built on different evidences. This operator has the same meaning as the conjunction operator in the Josang's model.

Considering the referrals network approach previously presented in Singh et al. [2001] and solely based on the Dempsters rule combination operator adapted to this environment, they fully describe an agent local decision model for selection of a transaction partner. In order to keep the referral graph restricted (because longer the referral chain is, less reputed is the obtained information) they introduced a depth limit of the referral graph.

They extended their previous experiments to a bigger number of agents (100 to 500), keeping the same vector-based information space for the expertise and the same average number of neighbours. They introduced a new parameter in the experiments: the cooperativeness factor an agent, after selected, might accept to perform a transaction with a certain degree. Computing the overall reputation of the agents in the simulated experiments, they reached the conclusion that overall reputation stabilizes to an equilibrium value. They also simulated the behaviour of a single agent who at the beginning was very cooperative thus gaining a very good reputation. After that, if its cooperativeness factor was reduced to simulate the abuse of having a high reputation, it was proved that its reputation decreased rapidly.

The experiments of Yu and Singh are valuable especially for P2P-based grid communities as they demonstrated formally that the predicted informal behaviour of an agent will really happen.

5. Agent-based approaches

Generally, the agent research community sees the agent paradigm as a good formalization for a wide variety of distributed systems, including grids, semantic web, pervasive computing and P2P [Huynh et al., 2006]. The most important property on which they base their discourse is the openness propriety of multi-agent systems, the fact that the agents are self-interested, proactive, know only a local part of the acting world and no central authority restricts the behaviours of all agents. This section will review the main reputation models developed by agent research community.

SPORAS and HISTOS

The systems proposed by Zacharia and Maes [2000] were one of the first attempts to build a reputation-based system to overcome existing trust problems in e-commerce on-line applications. Their ideas were incorporated in later reputation-based trust models.

First, they proposed the SPORAS system, based only on direct transaction ratings between users. Users rate each other after a transaction with continuous values from 0.1 to 1. The ratings received for a user are aggregated in a recursive fashion, obtaining a reputation value that scales from 0 to 3000 and a reputation deviation to assess the reliability of the reputation value. The recursive formula for updating the reputation is based on the following principle: users with very high reputation will experience much smaller rating changes after each update and ratings are discounted over time. The time discount model was further used in FIRE [Huynh et al., 2006].

HISTOS takes into account also the social network created between users through performing transactions. The reputation model for user A_0 from user A_i point of view takes in the consideration all paths on the social network graph between these 2 users. Only paths with positive (greater than 0.5) ratings are considered. As a rating is more far away from the user under discussion, its influence to the total social network rating is lower. The same kind of social network was used after that in the approaches of Despotovic and Aberer [2005]; Sherwood et al. [2006].

When evaluating SPORAS and HISTOS, the authors reported better results than the classical eBay and Amazon approaches. Although, their results were outperformed by more recent studies.

REGRET

Sabater and Sierra [2001] propose a model, named REGRET that considers the following dimensions of the reputation: *the individual dimension*: which is the direct trust obtained by previous experience with another agent, *the social dimension* which refers to the trust of an agent in relation with a group and *the ontological dimension* which reflects the subjective particularities of an individual.

Their model focuses on SLAs between two parties, with several variables under interest. Each agent stores a local database with impressions regarding the accomplishment of an agreed value of a SLA. The impression values are marked with time stamps, are continuous and might be positive, negative or neutral.

The subjective reputation of an agent with respect to another agent is computed against a pattern of SLA possible variables and takes into account the impressions stored in the local database weighted with a time discount factor. The reliability of the subjective reputations depends on the number and the variability of the impressions used to compute the reputation. For assessing the individual dimension, the above-described subjective reputation is computed.

For assessing the social dimension, first, the agent aggregates its subjective reputation for the agents members of a target social group. This is the assessment of the agents previous experience with a target group. Second, the subjective reputations of all agents in the same group with our agent are aggregated to obtain the group subjective reputation for a target agent. Third, an overall subjective reputation between groups is obtained by aggregating all subjective reputations between agents belonging to the groups under discussion. The reputation for the social dimension is obtained by aggregating all 3 components described above, including the individual dimension as a 4^{th} component. In all aggregations, weighs are used to reflect the importance the agent puts in one or another component of the aggregation. These weights might change during the agent lifetime

The ontological dimension is computed considering the internal ontological model an agent has with regard to a service. Therefore, one agent might be a good seller if he delivers on date, at an agreed product price with a certain quality. The ontological knowledge of an agent is composed by the aggregation structure between variables and their corresponding weights. To compute the subjective reputation for the ontological dimension, the agent aggregates the individual subjective reputations for the variables part of the structure of a desired variable in a SLA.

This model is simple and allows one to express easily the reputation for the individual experience and the group-related reputation and to compose services by aggregation. The service composition is a well-desired property of grid systems. Besides the internal agent database, another database is required at the level of a group in order to store the group-based reputation. This is a mean of centralization. Although the authors do not say by which mechanism one agent is said to belong to one group this is viewed as a drawback in Suryanarayana et al. [2004], with regard to grid technologies, one might consider the nodes organization as being the group of that node. The belonging of a node to a group is, therefore

resolved by authentication in grid systems.

Another drawback in the opinion of Suryanarayana et al. [2004] is the lack of referrals traffic. This is indeed a drawback in P2P approaches, as only part of the social information (the one between involved groups) is considered when assessing the general trust, but in classical grids this could be an advantage, as is hard to imagine that a node in an organization will easily inquire another node in another organization for a reference. Also, the lack of the referral queries decreases the bandwidth cost. The composition of a group reputation by aggregating the individual members reputation might represent a mean of cheating, if most members of a group are unfair ones. But in classical grids one can assume with a high likelihood the good intention of the participant nodes.

Overall, the REGRET system could be of interest for classical grids as allows a way of aggregating the reputation at the level of a group and approach the service composition, which is a particularity of grid systems.

Ramchurn et al. [2004b]

Taking the assumptions of Sabater and Sierra [2001] as valid, Ramchurn et al. [2004b] further refine the system by detailing more on the terms of a contract (service level agreement - SLA). Their intention is to build a trust model to be directly used in negotiation of a contract. As they see the negotiation process as a successive exchange of offers and counter-offers, they argue that a trust model can short the length of the negotiation and can assure better negotiated values.

The premises of the model are the following: the whole society is composed by groups of agents and each agent is part of one group. Some power-based relations exist between groups. Two agents negotiate on a contract made by several issues, for each issue the negotiation would establish a common accepted value. Agents get some utility after the execution of a contract. Each partner in a contract should have some expectations about the outcome values of the issues. In the environment, all agents must fulfill some societal rules - common to all agents, group rules - common only to agents in a particular group and institutional rules - coming from the interaction environment in which 2 agents negotiate and execute the contract. Each agent stores a history of the agreed contracts and the context (made by the rules) at the time when a contract was negotiated. The trust model is composed by two components: *confidence* - accounting for the direct trust (obtained only by the agents experience) and *reputation* - accounting from

the trust obtained from the social environment.

The confidence of an agent in an issue x handled by another agent is a measure of the certainty which allows the first agent to expect a given set of utility deviations to be achieved after the second agent will fulfill the contract. Bigger the confidence is, smaller the expected deviations are. Confidence can be bad, average and good, each of these linguistic label being associated a fuzzy utility function that maps utility deviations in the set $[0, 1]$. The confidence levels for an agent with respect to a contract issue is evaluated from the history of the past interactions, by building a probability density function of the agents utility variation. They employ a similarity function between contract values in order to filter out cases from the history which are not relevant to the actual negotiation context. With this approach they tackle the problem of an agent performing well in a long history of small transactions and after that, cheating in a big and very valuable one transaction [Resnick and Zeckhauser, 2002].

Regarding the reputation, they have a similar view as in Sabater and Sierra [2001]. They do not consider the problem of obtaining the reputation from the social environment, assuming that some method exist for getting it (like asking for referrals or the existence of a central reputation service for the group). The reputation measure is continuous, between $[0, 1]$ and reflects the first agents view about a second agent reputation in handling an issue of a contract with respect to a qualifying confidence level. The group reputation is aggregated as in the REGRET model, but considering a bigger weight for more powerful groups. Reputation measure is useful for an agent without prior transaction experience as it can base its negotiation process on it. Confidence and reputation are aggregated in order to obtain the final trust model.

The model principles do not differ too much from the one of Sabater and Sierra [2001], but it has the advantage of entering more in the details of the establishment of a SLA. It can be worth for the grid community, as the authors show how trust can be incorporated in the SLA bilateral negotiation process by permitting the adjustment of the proposed values for the issues of a contract in a more reliable way. Even more, when the trust that the negotiation partner will supply the agreed values in the contract is low leading to negative utility expectations, the model shows how an agent can further require more issues in the contract (as a new quality certification) in order to secure a positive utility

expectation.

FIRE

In the conception of Huynh et al. [2006] a trust model has to (1) take in consideration a wide variety of information sources, (2) the agents should be able to evaluate the trust for themselves (distribution and local control) and (3) the trust model should be robust to lying. They address the first 2 requirements, building a trust model based on 4 different types of trust: (1) interaction trust, (2) role-based trust, (3) witness reputation and (4) certified reputation.

The interaction trust is built considering the previous agents experience, as in the REGRET model. The ratings of a previous transaction are continuous, selected from $[-1, 1]$, only the last H ratings with regard to an issue and another agent are stored in the local database and when aggregating the previous ratings, a time discount function is employed. The role-based trust models the trust resulting from the role-based relationships between 2 agents (e.g. owned by the same company, the relationship between a service provider and its users etc). They propose some rules in order to assess the role-based trust. These rules are of the following form: if 2 roles are considered, a rule expresses the expected performance between agents belonging to these 2 roles and the confidence in the above-assessed expectation. The witness reputation is obtained from the social network of the agent, following a referral process like the one proposed by Yu and Singh [2002]. Therefore, queries are required to be propagated through the network in order to compute the witness reputation, which implies a higher bandwidth cost. The certified reputation of an agent consists of a number of certified references about its behaviour on a particular task. Each agent stores its own certified reputation like the references one has on her resume, and when other agent wants to see them, the agent makes its references available. As one agent will reveal the references its has about its previous tasks, it will have the incentive to present only good references, therefore it makes sense to store only the best reference obtained after fulfillment of a transaction.

To obtain the trust value for an agent, one has to aggregate each piece of reputation mentioned above. The authors propose to weight each component as to reflect the emphasis the model puts assigns for each of the information sources above. The weights are normalized. Each trust value is accompanied by a reliability value which, in turn, is composed of two measures: (1) a rating reliability

computed on the basis of the weight given for certain component, measuring the quality of the reliability and (2) a deviation reliability measuring the volatility of rating values and therefore, the certainty of the accomplishment of an agreed SLA.

They showed that each component of the model adds an improvement in how reliable and fast an agent finds its partners in transactions. More, they compared the model with a centralized approach (which is supposed to perform better as the whole amount of information is available in one central point) and demonstrated comparable performance levels.

We think that this model is the most complete one from the agent research point of view, combining the advantages of the previously described models of Sabater and Sierra [2001] and Ramchurn et al. [2004b]. Only few additional costs are involved, as more model components require more storage and the witness reputation requires a bandwidth cost.

ART Testbed

With the intention to unify the research with regard to reputation-based trust, a group of researchers from several universities launched the ART Testbed competition, supplying with a testbed for unifying the experiments related with reputation models [Fullam et al., 2004, 2005]. The first edition of the contest took place during AAMAS 2006 conference in Hakodate Japan with 14 registered agents, but the idea emerged in 2004 and get contour during spring 2005 with the papers of Fullam et al. presented at AAMAS 2005.

The testbed [Fullam et al., 2004] provides an environment for a limited number of competing agents 6, which have limited expertise in providing some services (evaluation of paintings), and which have to gain as many utility as possible (in terms on money) by performing the service during several rounds of the game. Agents can respond to a service request and might be engaged in exchanging opinion and reputation information. Opinion information regards the agents opinion about the value of a service and the reputation information regards the agents trust in a third agent. Services are assigned to agents by the simulation engine. Therefore, the agents should concentrate only in the social-based reputation model. With this respect, the testbed is valuable as it provides a mean of experimentation for modeling the trust obtained by direct experience and referral trust obtained by gossip through the social network.

Till now, some papers ([Fullam and Barber, 2006; Kafali and Yolum, 2006; Sen et al., 2006]) were already being produced based on the testbed. But, instead of achieving the goal of experimenting existing strategies in a unified world, these papers focus on the specificity of this environment.

Kafali and Yolum [2006] add a new factor to the reputation model: the self-confidence of the agent, as being the number of times an agent is asked to produce a reputation or an opinion. In their experiments they used agents equipped only with a direct trust model (based on the past transaction experience) and on a mixed model combining direct trust with reputation-based trust. An agent has also to consider its strategy when responding to reputation requests. An agent might respond sincerely to all reputation requests, thus being recognized as an expert and allowing other agents to gain more or might consider to respond only to those agents who performed sincerely in a previously reputation exchange. They found that the most beneficial strategy is to consider the reputation-based trust as part of the trust model and to respond sincerely to all reputation requests.

Sen et al. [2006] investigates the existence of cooperation opportunities as part of the testbed setup. They argue that a trustful behaviour should lead towards cooperation between individuals supplying complementary expertise for the overall long-term goodwill of the community. They demonstrate that in the actual environment setup, agents do not have incentive to cooperate on the basis of trust and they propose an improvement in this direction: to change the client share function. They also show by experimentation that such a setup based on trust management can lead for the cooperation between self-interested agents and conclude that an effective trust management scheme should (1) allow agents to be inclined to help someone that has a potential to provide help, (2) allow comparisons between different cooperation costs, (3) be able to flexible adjust inclination to cooperate based on the current work-load and (4) be responsible to changes in types of tasks and types of expertise in the population.

Fullam and Barber [2006] see reputation exchange as a mean on learning the trustworthiness of the agents. They apply the q-learning method as decision support. In this method, each agent is rewarded for each action it takes. Therefore, rewards are assigned for requesting and providing opinions and for requesting and providing reputation. The opinion and reputation values are selected according with the actual rewards an agent possesses. Their experiments show that learn-

ing agents gains more than non-learning or cheating agents, while it seems that the reputation model has only a little influence to the overall behaviour of the learning agent.

The novelty of this approach is the fact that the trust and reputation profile of the agents in the society is memorized in the form of related rewards. These rewards replace the well-known trust and reputation measures. Their approach is more a competing game-theoretical one. They are not concerned about the overall gains of the game or about the total welfare produced, but rather about the agent who will win the game.

Although one objective of the testbed was to provide a mean of experimentation for reputation methods, it seems that only very few experimentation were pursued on the testbed. Instead, authors focused on its game-theoretical property, trying to win the game rather than to observe the behaviour of a particular already developed reputation model. The paper of Sen et al. [2006] revealed some weaknesses of the testbed, from the agent research perspective. From the grid point of view, we can say that the testbed is not too valuable, as it can accommodate only a very small number of participants and the total length of a game do not allow building large history of transactions. More, the testbed focuses only on direct trust obtained by experience and indirect trust obtained by referrals, the other existing types of trust being not present in the testbed. The testbed does not allow trust aggregation, as in the model of REGRET [Sabater and Sierra, 2001], nor SLA negotiation as in the model of Ramchurn et al. [2004b] Therefore, its suitability for evaluation, with respect to grid research is very limited.

6. P2P approaches

In P2P systems, one main concern is the identification of malicious peers that provides misleading services. Trust models might prevent such behaviour and might improve the reliability and fault tolerance of the system. In a P2P approach, the challenge is how to aggregate the local trust values without a centralized storage management and facility. Beside, two kinds of questions are addressed by P2P approaches: what trust metric should be considered and how to store reliable and securely the trust values across the network.

P2P approaches are more suitable for fully decentralized grids, like desktop

grids, which come closed with P2P. Regarding their suitability for classical grids, they are quite far from the classical grid problems like SLA and QoS negotiation, or virtual organization management. But, as we will see, ideas from P2P approaches were considered by the grid community, allowing those ideas to be improved by some degree of centralization.

Gupta et al. [2003]

Gnutella-like P2P file sharing systems are among the most popular P2P networks. They are fully decentralized and unstructured and file sharing is their objective. In [Gupta et al., 2003], Gupta et al. proposes a reputation system to track back the past behaviour of users and to allow drawing up decisions like who to serve content to and who to request content from. They base their system on the internal properties of such a network, where the most important activities are content search and content download. One objective of the proposed reputation system is to give an idea about the level of participation of the peers in the system. The reputation system proposed by Gupta et al. [2003] is a transaction-based one, rather than the user-based approach of TrustMe [Singh and Liu, 2003], described in the following subsection.

In this model, the reputation of a peer depends on (1) its behaviour assessed in accordance with the contribution of the peer to content search and download and (2) its capability expressed in terms of processing power, bandwidth, storage capability and memory. Each peer in the network gets credit for (1) processing query response messages, (2) serving content and (3) sharing hard-to-find content in the network. Content serving and sharing hard-to-find content are assessed based on the quality of the service provided (in terms of the bandwidth and file size). For each download, a peer reputation is debited with a similar amount as for serving the same content. The reputation score is simply a summation of the receiving credits with or without deducing the debits.

Each peer could maintain and compute its reputation locally. But, because there is a misbehavior threat with regard of this operation, a reputation computation agent (RCA) is provided for the P2P network with the goal of keeping track of transactions and of the credits and debits that flows in the network. A peer might choose to participate in the reputation system then it will need to cooperate with the RCA, or might stay apart of the reputation system in this case its reputation is minimal (0). The RCA maintains a transaction state of

the system keeping track the full list of transactions and points to be granted for those transactions for a period of time. Each peer communicates with the RCA based on the classical public key cryptography exchange mechanism. After each transaction, each peer reports the transaction to RCA. From time to time the peers contact RCA for being granted with credit for their transactions. The RCA is a central point of failure only for the reputation management scheme. Therefore, the functionality of the P2P network will not be affected if the RCA fails, as it only adds with a supplementary functionality.

The system is simplistic, but covers well the properties of the target P2P network and does not interfere with the standard usage of a Gnutella-like network. Although some misbehavior is still possible as peers might report incorrect transaction details, the system tries to reduce the incentive of multiple identities because a new coming peer always receives no reputation. Some experiments were reported, showing the effectiveness of the reputation system.

This reputation system might have some importance for grid research as it presents a reputation scheme that gives score for desired behaviour and penalizes undesired one therefore, pushing toward cooperative behaviour. Also, it shows how issues part of QoS delivered can be included in the reputation.

TrustMe

TrustMe [Singh and Liu, 2003] is another approach for decentralized and unstructured P2P networks. Rather than the approach of Gupta et al. [2003] which is a transaction-based one, TrustMe is a user-based approach, adopting the principle of obtaining references about a peer, before engaging in a transaction with that peer. Broadly, TrustMe functions in the following manner: each peer is equipped with a couple of public-private key pairs. Trust values of a peer (B) are randomly stored at another peer (THA) in the network. Any peer A interested in the trust value of a peer B broadcast the query on the network and the THA peers replies this query. Based on the received trust value, peer A decides to enter or not in interaction with peer B. After interaction, peer A files a report for peer B indicating the new trust value for B and therefore, THA can modify the trust value of B accordingly. TrustMe uses a smart public key cryptography mechanism to provide security, reliability and accountability. It is assumed that somehow, peer A updates the trust information for peer B and broadcast back this information to its storage located at peer THA.

TrustMe lets free option for selecting the trust measure and focuses on developing a secured message exchange protocol for protecting the information and its sources in the network. Some properties of their proposed protocol are: persistence, no central trusted authority needed, small decision time and ease of contribution. It is out of the scope of this paper to develop the details of message exchanges protocol in TrustMe. But, it is worth for consideration as an alternative way of enforcing trust in a decentralized P2P network.

Comparing this approach with the one of Gupta et al. [2003], the bandwidth cost is increased, as each peer has to deal also with requests relating reputation besides its usual tasks for responding to search and download queries.

EigenTrust

According to Kamvar et al. [2003], the following issues are important in P2P reputation system: (1) self-policing: no central authority should exist and the peers should enforce the ethical behaviour by themselves, (2) anonymity: peer reputation should be associated with an opaque identifier, (3) the system should not assign profit to newcomers, (4) minimal overhead and (5) robust to malicious collectives of peers.

Their approach is based on the notion of transitive trust: a peer i have a high opinion of those peers who have provided it good services and therefore, peer i is likely to trust the opinions of those peers. The idea of transitive trust leads to a system where global trust values correspond to the left principal eigenvector of a matrix of normalized local trust values.

Kamvar et al. [2003] considers that each peer stores locally its trust values for the rest of the peers [Kamvar et al., 2003]. They do not enforce a method for obtaining these trust values, but they suggest the trust values could be obtained by evaluating each previous transaction between peers thus being a form of direct trust. Each peer normalizes these trust values obtaining values in the interval $[0, 1]$, 1 being assigned to the most trusted peer. In order to obtain a global view of the network, as in Yu and Singh [2002], each peer can ask referrals from its neighbours regarding a third peer. The received trust values can be aggregated using the local trust values for the neighbor as weights. Therefore, using one set of queries that is investigating the neighborhood graph on a distance of 1, a peer can obtain a trust vector including witnesses of first order. Iterating and querying the neighbours of the neighbours, the global trust vector becomes

much refined. Kamvar et al. proved that by further iterations, the global trust vector converges to a value that is unique for the network and is the left principal eigenvector of the initial matrix of normalized trust values [Kamvar et al., 2003]. Therefore, by a repeated query process, each agent can obtain the global trust vector, while still storing locally only its own trust values regarding the rest of the peers. This model has also a remarkable probabilistic interpretation, as a peer might interrogate its neighbours with the probability given by the neighbors local trust value. In order to make the model more resistant to collusion, they propose to consider the founders of the network as a-priori trusted nodes and at each iteration step, to take a part of the trust as being the trust given by these nodes. Addressing the distribution of the storage of the data, the paper lets each node to store also its global trust number part of the global trust vector, besides the normalized trust values. Doing this, the initial a-priori trusted nodes get lost in the network anonymity, making the model more reliable.

Kamvar et al. [2003] addresses also some issues which are specific to P2P architectures and are not in the scope of trust management, as how to avoid that a peer to wrongly compute its global trust value. A replication scheme is proposed, allowing each peer to compute the global trust value for other peers in the network.

Regarding the usage of the trust values, they propose to select the peer who will supply a service on a probabilistic basis, taking the selection probability as being a mixture between the global trust value of the peer offering the service and the local value stored at the requesting peer regarding the peer who supplies the service.

Doing some extensive experiments, they showed a good performance of the trust model in the P2P setup. Although, they could not totally reduced the failure rate in the system, but the improvements are significant.

PeerTrust

PeerTrust [Xiong and Liu, 2004] is based on 5 important parameters contributing to a general trust metric. The 5 parameters considered are: (1) the feedback a peer obtains from other peers, (2) the feedback scope counted as the number of total transactions that a peer has with other peers, (3) the credibility factor for the feedback source, (4) the transaction context factor discriminating between mission-critical and non-critical transactions and (5) the community context fac-

tor for addressing community-related characteristics. In fact, as revealed by their general trust metric formula, the trust metric for a peer is composed by the community context factor metric and the weighted satisfaction received for previous transactions.

The weighted feedback received for previous transactions internalizes the first four information sources mentioned above. Regardless of the feedback scheme used by the peers, the feedback should translate into a continuous $[0, 1]$ numerical satisfaction measure, accounting for the first 2 information sources. For assessing the credibility, a first choice they propose is to use recursively the existing trust values of the peers, building an averaged TVM metric. The second choice is to construct the credibility measure from the similarity of satisfaction vectors collected from other peers that interacted with both peers involved in a transaction. The transaction context factor could be in fact a time decay weighting function, allowing that more recent transaction to have a bigger influence. The community context factor can have a very big importance in the model, and its main intention is to provide with a way of convincing peers to give feedback for past transactions. Therefore, they propose as a measure the proportion between the number of the transactions for which a feedback is given and the total number of transactions of that peer. Regarding the distribution of the trust model, each peer has a trust manager that is responsible for feedback submission, for trust evaluation and a database that stores a portion of the global trust data. Several distributed algorithms are proposed for computing the various formulas required by the trust model.

They performed some simulation over several P2P setups with varying number of peers in order to find the effectiveness of the proposed formulas and algorithms. They also considered a defective behaviour of a part of peers. They concluded that the similarity-based approach for measuring the credibility is more efficient than a recursive trust based approach is a setup with malicious peers. When trust-based peer selection is employed in a collusive approach with the similarity-based measure for peers credibility, better results and bigger transaction rate is obtained in comparison with a standard setup without trust-based peer selection.

The importance of the paper is the demonstration that a trust-based mechanism for partner selection in a transaction is worth for consideration in a P2P approach. Also, they demonstrate that usage of 3rd party information for building

credibility (reputation indirect trust) is much valuable that only the own-existing experience. Rather than based on own evaluation of the experience, the model bases on feedback, taking its inspiration from eBay.

With regard to the usage of the model in classical grids, not too many things can be said, as the model does not tackle the problem of QoS and SLA negotiation. The trust measures developed part of the model could be of interest as they proved to be effective is a P2P approach. They also do not tackle the problem of peers belonging to different organizations, which is of interest in classical grids. Although, the model has a great importance regarding desktop grids, as it uses a full P2P approach.

P-Grid

In P-Grid, Aberer and Despotovic [2001] see reputation as an assessment of the probability that an agent will cheat. To compute reputation they use data analysis of former transactions. Their trust is binary; an agent can perform a transaction correctly or not. They consider that usual trust exists, and therefore, they disseminate only dishonest information as relevant. They name this information as complains. Therefore, agent p after detecting the malicious behaviour of agent q will store a complaint $c(p, q)$. The total trust of an agent p is defined as the number of complains the agent p stores multiplied with the number of complains about agent p stored by other agents. High values for this trust value indicated the fact that the agent is not trustworthy.

The global trust model is very simplistic and in this approach the main challenge is to store complains in a distributed manner in the network. The P-Grid solution is selected. A P-Grid is a virtual binary search tree, each leaf in the tree being associated with a node from the network. Each node stores data items for which the associated path is a prefix of the data key and also some routing information for directing a search to a complementary node. This P-Grid structure supports 2 operations: insertion of a new node with its related information and query for complains data about an agent. Search is done in $O(\log n)$ time and the storage space required at each agent scales also with $O(\log n)$.

For insertion of a new node, the same insert method is replicated for a number of times, chosen according with the supposed proportion of cheating agents. For locally computing the trust, an agent asks several queries for the same data and after that, she aggregates the received responses, according with the frequency a

witness agent is found. The decision regarding whether an agent is trustworthy or not is chosen according with the following heuristics: if an observed value for complaints exceeds the general average of the trust measure too much, the agent must be dishonest.

They evaluated their trust scheme on a population of 128 agents, with different number of cheaters in it and a big number of interactions (6400 or 12800). Good quality for the trust measure is obtained, and this quality can be increased only by increasing the data replication in the P-Grid. The scheme has also the quality of distinguishing well the cheating agents. Therefore, the advantage of the method is that it allows taking decisions regarding peers interactions in an increased number of cases, increasing the reliability of the P2P network.

The method is quite suited for P2P approaches and also for decentralized desktop grids. With regard to a standard grid, anyway, the usability of the method is under question, as it does not address the QoS and SLAs and not also the virtual organization formation.

NICE

The NICE approach [Sherwood et al., 2006] targets a specific P2P network implementation, the NICE1 platform. In their view, the trust value of a node B at a node A is a measure of how likely the node A believes a transaction with node B will be successful. They adapted the idea of the social network described in the agent-based approaches to the structure and the security requirements of a fully decentralized P2P network, equipped with a PKI infrastructure. Each agent comes to the system with a pair of public and private keys and the messages are signed by the peers who are creating them. Therefore, after each transaction between a peer client A and a servant B, the peer A generates a cookie with its perceived feedback (trust value) for the transaction. Trust values scales from 0 to 1. Peer A sends the cookie to B and peer B can store the cookie as a reference of its effectiveness in other transactions. Peer B can decide which cookies to store and how long to store such a cookie. More, each peer could posses its own algorithm for updating and storing the trust values it receives from transaction partners.

When a peer A deliberates to enter a transaction with peer B, a cookie might exist between A and B and in this case, this cookie contains the trust peer A has for B. Or previous transactions did not already exist or were discarded. In

this case, A will ask its partners about having cookies for B and the partners will continue to spread the request into the network till a path between A and B is established. As a response to its request, peer A will collect the cookies that link it to B and therefore, will have the graph structure of the social network. On this graph structures paths between A and B are evaluated either by selecting the minimum trust value on the path or by multiplying the trust values. Therefore, the strongest path can be selected. Refinements mechanisms are presented with regard to generating cookies requests. One of them is to allow users to store negative cookies. It is obvious that after a defective transaction, when peer A will generate a cookie for peer B with a low trust value, peer B will simply discard the cookie, as it does not help him. But instead, peer A can retain the cookie as a blacklist, and never entering transactions with peer B.

Experimenting with the system in various setups, the authors proved that the method scales well. Allowing each user to store a maximum 40 cookies and the outdegree (number of peers that receives the same message) of a cookie query to 5, they showed that querying at most 3 nodes in depth is enough to obtain a good representation for the social network. The total number of peers varied from 512 to 2048. When considering also malicious peers in the system (peers that do not follow the NICE trust protocol), a robust cooperative group emerged in the system. As they demonstrated, number of trust-related queries that are forwarded into the network is kept low; therefore, the total bandwidth overhead is minimal. As cookies are small and a peer does not have to memorize too many cookies, the memory requirements are kept also reasonable.

This approach shows how ideas from multi-agent research can be successfully employed in P2P computation. As the NICE is concerned with resource bartering, this environment comes closer to a fully distributed and decentralized grid.

7. Incentive compatible approaches

As we have seen in the previous sections, trust can be obtained both from direct interactions and via a third party source. After a transaction is finished, agents have to report about the result. Most studies assumed that agents report truthfully such information (eBay, Amazon, [Abdul-Rahman and Hailes, 2000; Huynh et al., 2006; Jøsang, 1999; Kamvar et al., 2003; Kerschbaum et al.,

2006; Ramchurn et al., 2004b; Sabater and Sierra, 2001; Singh et al., 2001; von Laszewski et al., 2005; Wishart et al., 2005; Yu and Singh, 2002]). When considering indirect third party sources to account for the reputation of an agent, again, the third party agent might lie and report incorrect information.

Some of the studies listed before analyzed in some extent the agents truthfulness and how robust the proposed reputation schemes are to such attacks. In this section we will shortly list these results and therefore, we will present the incentive-compatible reputation mechanism of Jurca and Faltings [Jurca and Faltings, 2003], whose design was guided exactly by these considerations.

Despotovic and Aberer [2005] experimented their system against liar agents and reported good results when the number of liars is low and there are enough agent interactions. But, the performance gets worse as half of the population lies and the number of direct interactions is reduced. Agents are let to deduce the misreporting probability from their direct interactions.

Regarding the ART testbed setup, Kafali and Yolum [2006] barely reported that playing honest when responding to reputation requests is the most beneficial strategy. Fullam and Barber [2006] did not investigate the effects of coordinated lying strategies.

In P2P systems, the designers usually do not allow peers to store theirs trust values and the storage model is distributed and replicated through the all network. Most of them consider this trust storage scheme as enough for protecting against lying nodes. In P2P systems like Gnutella, Gupta et al. [2003] recognizes an increased possibility of collusion when debits are not considered as part of the reputation measure. Cheating like reporting fake feedback is not possible in this setup, because the reputation points are uniformly given per transaction basis. In TrustMe [Singh and Liu, 2003], the authors designated a majority voting protocol in order to assure the reliability of the trust values communicated in the network. In PeerTrust [Xiong and Liu, 2004] the authors experimented with opportunistic cheating players but they only reported which of their proposed trust scheme performs better. In P-Trust [Aberer and Despotovic, 2001] the data replication scheme protects against lying.

Jurca and Faltings [2003] proceed with a game-theoretical approach when developing an incentive-compatible reputation mechanism. They argue that it is not in the best of an agent to (i) report reputation information because it

provides a competitive advantage to others; (ii) report positive ratings because the agent slightly decrease its own reputation with respect to the average of other agents and therefore, reporting negative ratings the agent will increase its own reputation. They base their model on the classical Prisoner Dilemma played iteratively. They acknowledge that an incentive-compatible mechanism should induce side-payments that make rational for agents to share reputation. These side payments are managed by a set of broker agents called R-Agents that buy and sell reputation information. The interaction protocol is as it follows: before a transaction, the agents select a R-Agent whom they ask about reputation. Each agent asks the R-Agent about the reputation of the partner and pays for this information. After finding out knowing the reputation of the partner, the agent can decide to engage in the transaction (play the game) or stay apart. If both agents decided to play the game, they enter a contract negotiation stage where they agree about the transaction terms and after that, they do the transaction and receive the payoffs for the transaction. From the payoffs, they can determine the behaviour of the partner in the transaction and submit a report to the selected R-Agent. After submitting the report, they will get a payment for this from the R-Agent. The agents also update their view about the effectiveness of the R-Agents regarding the reputation transactions. Payoffs obtained by transactions and by selling reports to the R-Agents are not interchangeable.

Regarding the payments an agent receives from a R-Agent, they selected the following payment scheme: if agent A reports about agent B behaviour and the report is the same as the next report received about agent B, in this case agent A will receive a positive payment for the report, otherwise nothing. They proved that in the case that the joint probability of lying inside the population is less than 0.5, the agents will be rational by reporting truthfully to R-Agents.

R-Agents are points of centralizing information in the system. It is possible that some R-Agents to have more accurate information than other R-Agents. Therefore, is important for usual agents to learn how to select R-Agents when requesting reputation information about transaction partners. A q-learning scheme is proposed for selection of the R-Agents, each R-Agent being selected according with the maximum expected reward value.

They experimented with this setup and showed that agents that use reputation information before engaging in a transaction accumulated much wealth

that agents that did not use reputation information. 40% of bad transactions were eliminated through the usage of the reputation incentive mechanism. More, introducing liar agents in the world, they showed that these agents finished by loosing money, while the trustful agents performed well.

The authors extended the model for pricing services in P2P networks [Jurca and Faltings, 2005a] and for improving the service level agreement in the web services world [Jurca and Faltings, 2005b]. In Jurca and Faltings [2005b], they considers groups of customers (like silver, gold and platinum customers) being serviced by providers and submitting binary feedback for the received QoS. The reputation of a provider is therefore the average positive feedback submitted by members of a customers group. Therefore, reputation is identical with the QoS delivered to a group of customers. The reputation mechanism also uses some trusted nodes that submit high trusted reputation reports. To assure that customers will report truthfully about the received QoS, as in the previous work, they consider side-payments for each valid report submitted to the reputation mechanism. Providers have the incentive to supply with the advertised QoS because some penalty payments are considered in the case they missed to accomplish the established SLA. The size of the penalty payments is computed taking into account the reputation of the provider.

These last papers ([Jurca and Faltings, 2005a,b]) are worth for consideration for the Grid community as they show directly how principles of rational behaviour from economics and game theory can be used to put incentives on the grid players to behave for the goodwill of the community. More, although the presented models have some degree of centralization, this is not a drawback in what concerns classical grids, as entity owners can behave as R-Agents for memorizing the reputation of the players.

1.3.2.3 Using reputation in grids

Up to date, there are several approaches of applying reputation models to grid systems. Reputation models can bring with more dependability in the grid by tackling the sabotage-tolerance problem or by improving the resource allocation and scheduling in grids. In either cases, the usage of reputation models affects the notion of trust in the grid computing environment, allowing the system to construct the *soft* version of trust. Sabotage-tolerance problem is specific to desktop

grids and there are several approaches with this regard. In classical grids, models described in the previous section were applied mainly in resource management with respect to virtual organization formation and evolution phases. In this subsection we will describe these approaches. The sabotage tolerance problem of the computational desktop grids will be developed in section **??**. This subsection will describe the approaches that employed reputation models in classical grids tackling resource management through virtual organizations. Service level agreements and quality of service negotiation are of particular interest.

GridEigenTrust

von Laszewski et al. [2005] exploits the beneficial properties of EigenTrust [Kamvar et al., 2003], extending the model to allow its usage in grids. They integrate the trust management system as part of the QoS management framework, proposing to probabilistically pre-select the resources based on their likelihood to deliver the requested capability and capacity.

They took the basic framework of EigenTrust and adapt it for grid requirements, resulting the GridEigenTrust model. First, to integrate trust in a QoS management, trust should be related to multiple existing contexts. If we discuss about grids, we need to address entities, organizations and virtual organizations. Considering 2 organizations which entities interact, a trust table will store the direct trust between the organizations, for each context of the transactions. The global trust between organizations at time t is computed by weighting the direct trust table entry with a time decay weight. The trust relationship of organization i for another organization j for a context c is obtained by aggregating the direct trust between these two organizations with the reputation of organization j, weighted with normalized values. The global trust or reputation of an organization j is computed by obtaining recommendations from a 3^{rd} organization and by aggregating the received recommendations with the direct trust values, applying the time decay function specific for the given context. This value is normalized as to scale to $[0, 1]$.

Considering a hierarchical organization of the entities, the trust of an organization will be computed based on the trust of belonging entities. The trust of a virtual organization will be computed based on the trust of the internal organizations. The updated trust of an entity is the weighted average between the old trust of the entity weighted with the time decay measure and the trust of

the organization to which the entity belongs to, weighted with the importance (grade) of the entity in the organization. A new organization that just joins the grid may be assigned a low trust or a trust with similar organizations, already part of the grid. The reliability trust of an organization could be obtained by normalized weighted sum of the direct experience and the global trust in that organization. To this weighted sum they also add the grade that users from trusting organization assign to entities part of the trusted organization.

These global reliability trust values are used as normalized trust values in the EigenTrust model, being therefore, used to compute by iteration the global trust vector of the virtual organization. As the P2P architecture of Kamvar et al. [2003] is no more of interest, a reputation service manager will perform all trust computation. The reputation service is composed by a data collection manager, a storage manager, a reputation computation manager and a reputation reporter.

The approach of von Laszewski et al. [2005] is one of the few from literature to propose a reputation service as a way to improve QoS management in grids. Although they present the design of the system, they do not present experiments in order to prove the efficiency of the approach.

PathTrust

PathTrust [Kerschbaum et al., 2006] is a reputation system proposed for member selection in the formation phase of a virtual organization. Because virtual organizations represent one of the main abstraction of the grid [Foster et al., 2001], we described PathTrust as a grid-related reputation system.

To enter the VO formation process, a member must register with an enterprise network (EN) infrastructure by presenting some credentials. Besides user management, EN supplies with a centralized reputation service. At the dissolution of the VO, each member leaves feedback ratings to the reputation server for other members with whom they experienced transactions. The feedback ratings can be positive or negative ratings. The system requires each transaction to be rated by the participants.

PathTrust arranges the participants in a graph structure similar with the one of NICE [Sherwood et al., 2006] or agent-based social networks [Singh et al., 2001]. Each edge in the graph is weighted with the trust between the nodes at the ends of the edge. This trust is computed by accounting the number of positive feedback let by participant i for participant j and subtracting the number of

negative feedback weighted by the report between the total positive feedback and total negative feedback participant i has submitted. If the report is less than 1 that is i submitted more negative feedback, then the weight is 1. The above trust value is normalized by the total number of transactions and therefore, it is less than 1. To distinguish between no transactions experience at all and some existing experience, the trust value is lower bounded by some small value (0.001). The weight of a path in the graph is the product of the weights of the edges that compose that path. As in NICE [Sherwood et al., 2006], for assessing the reputation between 2 nodes in the graph, the PathTrust algorithm selects the path with the maximum weight. Like in the EigenTrust [Kamvar et al., 2003] approach, the trust value is seen as the probability of selecting a participant from the list of possible alternatives.

They evaluated the PathTrust scheme against the EigenTrust algorithm and against attacks by reporting fake transactions in the system. It seems that with EigenTrust, a cheater can gain more profit than with PathTrust. The second test they performed was against random selection of participants. The results show that EigenTrust looses its advantage over random selection once cheating was introduced in the system. This loss occurs also with PathTrust, but is much lower. Therefore, to prevent cheating, the authors propose the usage of a transaction fee.

PathTrust is one of the first attempts to apply reputation methods to grids by approaching VO management phases. They approached only partner selection and did not tackled organizational aspects. Their model still lacks of dynamics, as the feedback is collected only at the dissolution of the VO. But, the advance in the field is given by the fact that ideas from previous research were successfully transferred in the area of virtual organizations and grids.

1.3.2.4 Conclusion

Grids pool together resources of various kinds and from various providers. Assuring a trusted computational environment is one of the fundamental requirements in Grid computing. Up-to-date, a lot of efforts were directed toward building trust using security mechanisms. But, as the Grids evolves in the direction of P2P computing and business usage, in the context of a fully transparency and automation at the level of resource-to-job assignments, reputation-based tech-

niques for building trust come into discussion. This paper reviewed the existing research in the area of reputation management, carried out in various fields of computing: Internet, e-commerce, agent systems, P2P and grids. We identified the most important properties a designer has to consider when approaching a reputation management system, depending on the context of applicability.

In general, models based on rational behaviour principles from economics as the one of Jurca and Faltings [2003] are worth for consideration as they allow nodes to behave autonomously and still to keep stability and goodwill in the society. Of course, the assumption that trust is a belief [Jøsang, 1999] and has some degree of uncertainty needs to be incorporated in the model. In the context of classical grids, centralized or semi-centralized approaches are still valid. One has to consider reputation aggregation in the context of virtual organizations, as in the approach of Sabater and Sierra [2001]. Other requirement to be considered in the case of classical Grids is the SLA and QoS negotiation. Models that emphasize on SLA [Huynh et al., 2006; Jurca and Faltings, 2005b; Ramchurn et al., 2004b] are worth for consideration.

For P2P systems and desktop grids, decentralized solutions are required. The approach of Zhao et al. [2005] reports goods results for failure detection with reputation mechanisms. One has to consider memory and bandwidth costs in such networks when devising a reputation management scheme because model distribution incurs such drawbacks. Some reputation management schemes reported good results with respect to these requirements [Gupta et al., 2003; Sherwood et al., 2006]. Including reputation acquired from the social network is valuable as some papers reported high trust induced in comparison with models using only direct reputation information [Xiong and Liu, 2004].

In the context of Grid systems, not too many reputation-based approaches are in the market. We can not recommend a reputation system as being the best of all Grid requirements, the design of the reputation mechanism being hardly dependent on the solution used for implementing the Grid middleware, how services are expressed in the Grid and how they are distributed.

In Grids, further research should concentrate on addressing resource selection and job allocation using algorithms that incorporate reputation of entities. Considering the virtual organization concept as the main abstraction of the grid, a reputation model should at least accomplish the trust aggregation and SLA and

QoS negotiation requirements. Regarding desktop grids, we think that job allocation can be improved with the usage of reputation models, mainly in the case of untrusted environments with high failure rates or big number of saboteours. Usage of reputation models can reduce the gap that currently exists between classical grids and desktop grids, making desktop grids trustable and allowing them to be used as the classical grids are.

1.3.3 A Model of Virtual Organizations

In order to support rapid formation of VOs, we use the concept of *virtual breeding environment* (VBE) [Camarihna-Matos and Afsarmanesh, 2003] adopted from the Virtual Enterprises community. A VBE can be defined as an association of organizations adhering to common operating principles and infrastructure with the main objective of participating in potential VOs. For this research, we have adopted the view that organizations participating in a VO are selected from a VBE, as illustrated in figure 1.2.

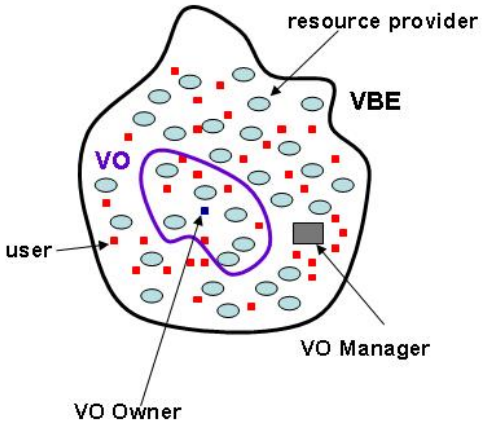


Figure 1.2: VBE and VO Models

Such organisations may provide services, represented by ovals, and include users that utilize VO services, represented by small squares. Organisations preregister to a VBE via the VO Manager component, including description of the services they are willing to share in a Grid and the list of potential users belonging to the organisation. When a user wants to create a VO, he assumes the role of VO

Owner and contacts the VO Manager with the description of the needed services. The VO Manager includes a service brokering component that suggests potential concrete services and their service providers. The VO Owner then selects a subset of these service providers and their services, and then defines the list of users for the VO.

The VBE can be seen as a market place where service providers are competing to participate in VOs and users within VOs are competing to use services. Reputation information about service providers can be used as a parameter for guiding the selection of VO partners. On the other hand, having reputation information about users could help service providers to implant tighter security mechanisms for accessing their services and the resources underlying them.

1.3.4 A Utility-Based Reputation Model for VOs

In this section, we develop a general utility-based reputation model for VOs, which will be used later to manage reputation in service-oriented VOs. Our reputation model is described in detail in Silaghi et al. [2007b]. The model was initially devised for service-oriented computing in grid systems and improves the models presented in the related work section 1.3.2.2.

Central to our model is the notion of an *organisation*. The set of all organisations is denoted by *Org*. We keep track of all VOs that have existed and use the set *VOId* to denote the set of all VO identifiers. The *entities* we want to keep reputation values for are defined as elements of the set *Ent*. An obvious restriction is that an entity must belong to an organisation. We are interested in some particular *issues of interest* associated to an entity; the set of all issues of interest is represented by *Issue*. The individuals that *consume* (use) the entities and qualify them are members of the set *Cons*. The sets *Ent*, *Issue* and *Cons* are considered type parameters that will be instantiated according to the domain we are interested in. Below we represent above types and their associations as functions, using the mathematical notations provided by the Z specification language [Woodcock and Davies, 1996].

$$[Org,\ VOId]$$

$$VOS : VOId \pfun \mathbb{P}\, Org$$

$$[Cons, Ent, Issue]$$

$$EntOrg : Ent \rightarrow Org$$
$$EntIssue : Ent \rightarrow \mathbb{P}\, Issue$$
$$EntCons : Ent \nrightarrow \mathbb{P}\, Cons$$

Notation $[Org, VOId]$ introduces Org and $VOId$ as basic types. VOS associates a VO with the set of organisations participating in it. Function $EntOrg$ associates an entity with the organisation to which it belongs to; $EntIssue$ relates an entity with its issues of interest; and $EntCons$ associates an entity with its consumers.

In our model, we assume the existence of *monitors* that deliver events indicating the current value produced by an entity for a consumer in relation to a particular issue of interest within a VO, at an observed moment of time. We represent an event as a tuple that contains of the following elements: the *time stamp* of the event, a *consumer*, an *entity*, an *issue*, $VOId$ and a k number of attributes, $Attr$. A trace corresponds to a sequence of events:

$$\begin{aligned} Event &== TimeStamp \times Cons \times Ent \times Issue \times \\ &\quad VOId \times Attr_1 \times \ldots \times Attr_k \\ Trace &== \text{seq}\ Event \end{aligned}$$

A utility function reflects the satisfaction of a consumer in relation to a particular entity. It relates an event with a numeric value indicating what is really received by the consumer:

$$utility : Event \rightarrow [0, 1]$$

The complete definition of a utility function is considered to be domain specific.

Utility functions are used to define the reputation of an entity in relation to a particular issue of interest from the perspective of a consumer.

$$[Cons, Ent, Issue]$$

$$rep\_eic : Time \times Cons \times Ent \times Issue \times VOId \rightarrow [0, 1]$$

$$\forall\, c : Cons, e : Ent, i : Issue, vo : VOId \bullet$$
$$rep\_eic(t, c, e, i, vo) = \frac{\sum_{ev \in Trace \upharpoonright \{(ts,c,e,i,vo,\cdots) \in Event\}} \varphi(t,ts)\, utility(ev)}{\#(Trace \upharpoonright \{(ts,c,e,i,vo,\cdots) \in Event\})}$$

where $\#s$ denotes the cardinality of sequence s and $s \upharpoonright A$ denotes the largest subsequence of s containing only those objects that are elements of A. $\varphi(t, ts)$ is a time discount function that puts more importance on events registered closer in time with the moment of computing the reputation. Reputation, *rep_eic*, is defined as the weighted average of the utilities obtained from all generated events so far; it is defined as a generic function parameterized by sets $Cons, Ent$ and $Issue$.

As a fitness measure for the above-defined reputation, we consider the reputation deviation *dev_rep_eic*. The reputation deviation shows how much the reputation varies in time and it evaluates the stability in the behavior of the VO members.

$$[Cons, Ent, Issue]$$

$$dev\_rep\_eic : Time \times Cons \times Ent \times Issue \times VOId \rightarrow [0, 1]$$

$$\forall\, c : Cons, e : Ent, i : Issue, vo : VOId \bullet$$

$$dev\_rep\_eic(t, c, e, i, vo) = \frac{\sum_{ev \in Trace \upharpoonright \{(ts,c,e,i,vo,\dots) \in Event\}} \varphi(t,ts) |utility(ev) - rep\_eic(t,c,e,i,vo)|}{\#(Trace \upharpoonright \{(ts,c,e,i,vo,\dots) \in Event\})}$$

Aggregating the reputation of an entity over all its consumers within a VO produces the reputation of the entity in the VO with respect to a particular issue of interest.

$$[Ent, Issue]$$

$$rep\_ei : Time \times Ent \times Issue \times VOId \rightarrow [0, 1]$$

$$\forall\, e : Ent, i : Issue, vo : VOId \bullet$$

$$rep\_ei(t, e, i, vo) = \frac{\sum_{c \in EntCons(e)} rep\_eic(t,c,e,i,vo)}{\#EntCons(e)}$$

Reputation function *rep_ei* is defined as a generic function paremeterised by sets *Ent* and *Issue*.

The reputation of an entity in a VO is then the aggregation of its reputation in each of its issues of interest within that VO.

$$
\begin{array}{|l}
\hline
[Ent] \\
rep\_e : Time \times Ent \times VOId \rightarrow [0,1] \\
\hline
\forall\, e : Ent, vo : VOId \bullet \\
rep\_e(t,e,vo) = \frac{\sum_{i \in EntIssue(e)} rep\_ei(t,e,i,vo)}{\#EntIssue(e)} \\
\hline
\end{array}
$$

The general VBE reputation of an entity is then the aggregation of its reputation in all VOs.

$$
\begin{array}{|l}
\hline
[Ent] \\
rep : Time \times Ent \rightarrow [0,1] \\
\hline
\forall\, e : Ent \bullet rep(t,e) = \frac{\sum_{vo \in \mathrm{dom}\ VOS} rep\_e(t,e,vo)}{\#\,\mathrm{dom}\ VOS} \\
\hline
\end{array}
$$

This last definition assumes that the domain of *VOS* represents the VBE; i.e. all the organisations participating in it.

Reputation deviation can be defined for all above-presented reputation deviation.

1.3.4.1 Properties of the Reputation Model

In this section we shortly discuss the properties of the reputation model presented above.

At the beginning we should note the usage of monitors to gather data for building the reputation. Most reputation models presented in section 1.3.2.2 are based on direct or indirect feedback collected from information sources with questionable reputation. As many VBEs (like Grids) supply with trustable monitoring services, using the data provided by those services seems to be a good alternative.

Next, central to the described reputation model resides the utility functions. Utility functions reflect the consumer perception about the delivered services. The utility function is an intrinsic evaluation of each consumer and might be difficult to construct it. We assume that somehow, the reputation manager builds the utility function of each consumer *before* the service to be delivered. For example, every user that joins a VBE and registers to the reputation management

service can pass a questionnaire used to further infer the utility function for that user. Knowing the utility functions before service delivery is a key feature of our model because it reduces the risk of cheating. If the consumer reveals out another utility function than its real one, it will end up with a service delivery associated with this wrong utility function, thus, will not benefit any more from the participation in the collaborative system.

When computing the reputation, instantaneous utility values associated with the events are weighted using a time discount function. Putting more emphasize on newer events represents a widely accepted approach in the reputation management literature [Huynh et al., 2006; Ramchurn et al., 2004b; Sabater and Sierra, 2001]. We recommend the following time discount function, also adopted by Huynh et al. [2006]:

$\varphi(t, ts) = e^{-\frac{t-ts}{\lambda}}$

λ is a parameter used to tune the importance of the newer events against older ones and its value is related with the time scale employed by the monitors for registering the events.

Figure 1.3 depicts the reputation for a service with one issue uniformly delivered in a variation band of 85% to 105% of the agreed QoS for the issue. On the top plot we depicted the cloud of the observed events. We considered 1000 time units. We can notice that at the begining of the experiment, after a short learning time frame, the reputation stabilizes itself around a value.

Figure 1.4 we considered a similar setup, but now, between time units 200 and 300 the consumer perceived a decay in the QoS. We can note the reputation recovers slowly after the dramatic decay in the QoS and on the long time, as the consumer gets continuously the same delivery patters as before the decay, the reputation converges to the initial value.

Reputation deviation can be used to decide in cases the service is delivered with a fluctuating quality. A reputation value accompanied with a smaller reputation deviation indicates a higher confidence in the expected value for the item under study. For example, normal distributed values around a central average are less fluctuating than uniformly distributed values around the same average. The upper part of figure 1.5 shows the how reputation varies in time for two patterns of QoS delivery explained above. In the lower part of the figure we can notice that the reputation deviation curve for the case of the normal delivery is situated

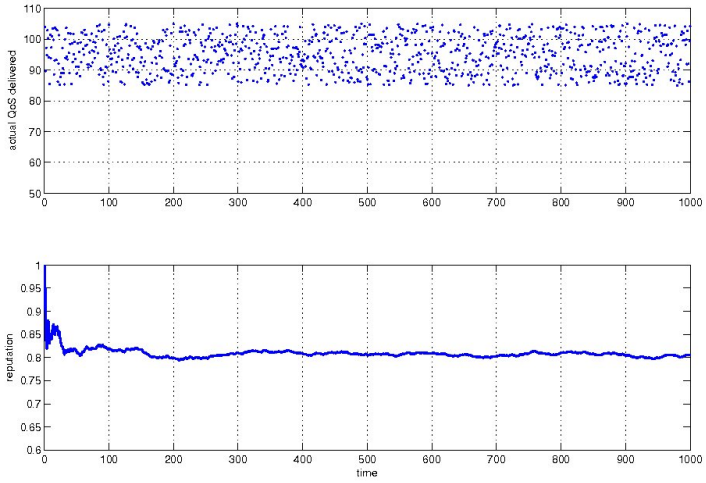


Figure 1.3: Reputation when the issue is delivered uniformly distributed in a variation band

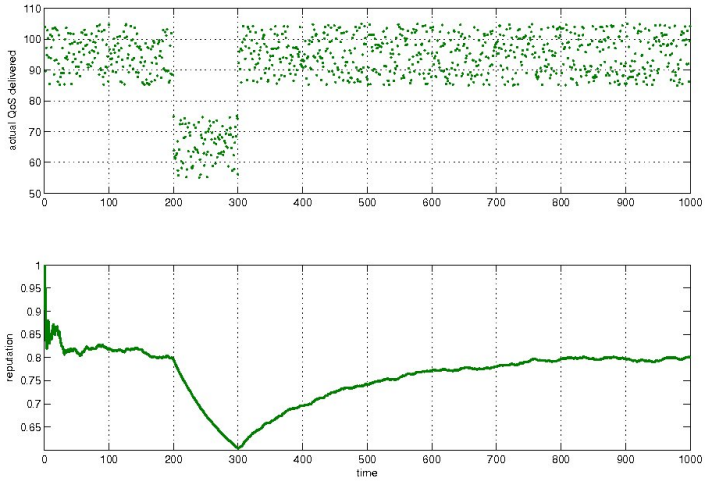


Figure 1.4: Reputation when there is a decay in the QoS delivery

below the curve for the uniform delivery of the QoS, indicating much confidence on the reputation of the first provider.

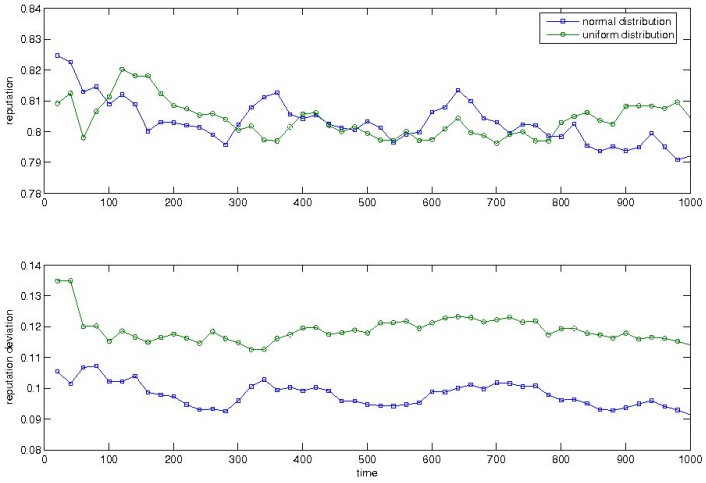


Figure 1.5: Reputation deviation used to assess the confidence in the reputation value

1.3.4.2 Reputation Management for VO Service Providers

Here we aim at maintaining reputation for organisations as service providers in a VO according to the Quality of Service (QoS) of the services they provide. In this model, *consumers* correspond to the *users* in a VO, denoted by $VOUser$, and *entities* correspond to the *VO services*, denoted by the set Srv. There are several options for selecting issues of interest. We can have either a fine granularity where each service level objective defined for a service can be seen as an issue of interest, or a coarse granularity where the whole QoS can be seen as a single issue of interest. For simplicity, we select the latter option.

Functions $UsersVO$ and $SrvVO$ represent the set of users of a VO and the services that an organisation provides to a VO, respectively.

$$UsersVO : VOId \rightarrow \mathbb{P}\, VOUser$$
$$SrvVO : VOId \times Org \rightarrow \mathbb{P}\, Srv$$

As we mentioned earlier, our model requires the existence of monitors capable of detecting variations in the QoS of each service and generating events to inform the reputation system about such variations. An event is then represented as a tuple denoting the current value of the QoS of a service being used by a user

within a VO.

$$Event == TimeStamp \times VOUser \times Srv \times \{QoS\} \times VOId \times \mathbb{R}$$

where QoS is a name indicating the QoS issue. In order to define the corresponding utility function, we introduce an auxiliary function indicating the Service Level Agreement (SLA) accorded between a VO user and a service provider for a particular service within a VO.

$$SLA : VOUser \times Srv \times VOId \to \mathbb{R}$$

The SLA function represents the *expected* quality of a service. It is used to define the utility (*satisfaction*) a user gets when consuming a VO service.

$$utility : Event \to \mathbb{R}$$

$$\forall (u, s, QoS, id, v) \in Event \bullet$$
$$utility((u, s, QoS, id, v)) = \begin{cases} 1 & \text{if } v \geq SLA(u, s, id) \\ \frac{v}{SLA(u,s,id)} & \text{if } v < SLA(u, s, id) \end{cases}$$

We can now define the reputation of a service using the reputation functions defined in the previous section. Here *Srv_rep_eic* denotes the reputation value given by a particular VO user to a service in relation to its QoS in the VO. *Srv_rep_ei* represents the reputation of a service taking into account its QoS in a VO; it is an aggregation of the reputation given by all users to the service in relation to the QoS issue of interest within that VO. *Srv_rep_e* denotes the reputation of a service in a VO. Finally, *Srv_rep* indicates the general reputation of a service. Note, since we have only one issue of interest, *Srv_rep_ei* and *Srv_rep_e* will be equivalent. All these reputation values are computed at a given time moment.

$$Srv\_rep\_eic == rep\_eic[Time, VOUser, Srv, \{QoS\}]$$
$$Srv\_rep\_ei == rep\_ei[Time, Srv, \{QoS\}]$$
$$Srv\_rep\_e == rep\_ei[Time, Srv]$$
$$Srv\_rep == rep[Time, Srv]$$

Using the above functions, we can now define the reputation of an organisation in a VO and its reputation in a VBE. The reputation of an organisation in a VO,

denoted by *Org_rep_VO*, is defined as the aggregation of the reputation of all the services it provides to that VO.

$$
\begin{array}{|l}
Org\_rep\_VO : TimeStamp \times Org \\
\times VOId \to \mathbb{R} \\
\hline
\forall\, o \in Org, vo \in VOId \bullet \\
Org\_rep\_VO(t, o, vo) = \\
\qquad \dfrac{\sum\limits_{e \in SrvVO(vo,o)} Srv\_rep\_e(t,e,vo)}{\#SrvVO(vo,o)}
\end{array}
$$

On the other hand, the general VBE reputation of an organisation is defined as the aggregation over all its VO reputations.

$$
\begin{array}{|l}
Org\_rep\_VBE : TimeStamp \times Org \to \mathbb{R} \\
\hline
\forall\, o \in Org \bullet \\
Org\_rep\_VBE(t, o) = \\
\qquad \dfrac{\sum\limits_{vo \in \{v \in \mathrm{dom}\ VOS \mid o \in VOS(v)\}} Org\_rep\_VO(t,o,vo)}{\#\{v \in \mathrm{dom}\ VOS \mid o \in VOS(v)\}}
\end{array}
$$

1.3.4.3 Reputation Management for VO Users

Next, we discuss how to maintain the reputation of users within a VO and within a VBE according to their usage of VO services. In this model, *users*, denoted by set *VOUser*, are seen as *entities* who could execute some pre-defined actions on services following pre-established policies. *Services*, denoted by set *Srv*, are seen as *consumers* that qualify users in relation to their actions. If a user attempts to execute an action that is not allowed by the VO policy, it will be given a bad qualification by the service that would be reflected in the user's reputation. In some sense, the model here reverses the notions of consumers and entities with respect to the model of previous section.

The set, *Action*, denotes the set of possible actions that can be performed on services. A policy indicates the set of actions a user is allowed to perform on the service in a VO. It represents the expected behaviour of the user.

$$
\begin{array}{|l}
policy : VOUser \times Srv \times VOId \to \mathbb{P}\, Action
\end{array}
$$

A penalty function penalises a user with a value in the interval $[0, 1)$ if he executes non-permitted actions

$$
\begin{array}{|l}
penalty : VOUser \times Srv \times VOId \\
\qquad \times Action \rightarrow [0, 1) \\
\hline
\forall\, u : VOuser, s : Srv, vo : VOId, a : Action \bullet \\
\quad (u, s, vo, a) \in \operatorname{dom} penalty \Rightarrow \\
\qquad a \notin policy(u, s, vo)
\end{array}
$$

Now, events are defined as follows:

$$
\begin{array}{lll}
Event & == & TimeStamp \times Srv \times VOUser \times \\
 & & \{Usage\} \times VOId \times Action
\end{array}
$$

where *Usage* is a name indicating the service-usage issue of interest. We assume the existence of functions, *policy* and *penalty*, that are used to define the utility that a service gets according to the actions performed by a user in a VO. These functions are domain-specific, and based on them, once can define the utility function as follows.

$$
\begin{array}{|l}
utility : Event \rightarrow \mathbb{R} \\
\hline
\forall (r, u, Usage, vo, a) \in Event \bullet \\
utility((r, u, Usage, vo, a)) = \\
\left\{ \begin{array}{l} 1, \text{ if } a \in policy(u, r, vo) \\ 1 - penalty(u, r, vo, a), \text{ if } a \notin policy(u, r, vo) \end{array} \right.
\end{array}
$$

We can now define the reputation of a user using the reputation functions defined in Section 1.3.4. Here *User_rep_eic* denotes the reputation value given by a particular service to a VO user in relation to the *Usage* of the service in a VO. *User_rep_ei* represents the reputation of a user taking into account its service usage in a VO; it aggregates the reputation of the user for all services he uses in the VO. *User_rep_e* denotes the reputation of a user in a VO; it corresponds to an aggregation of the reputation of all his issues of interest. Since we have only one issue of interest, *User_rep_ei* and *User_rep_e* are equivalent. Finally, *User_rep* indicates the reputation of a user in the VBE.

$$
\begin{aligned}
User\_rep\_eic &== rep\_eic[Time, Srv, VOUser, \\
&\quad \{Usage\}] \\
User\_rep\_ei &== rep\_ei[Time, VOUser, \\
&\quad \{Usage\}] \\
User\_rep\_e &== rep\_ei[Time, VOUser] \\
User\_rep &== rep[Time, VOUser]
\end{aligned}
$$

1.3.5 A Reputation Management System

Here we present the architecture of a VO that uses reputation management in order to facilitate the rating of both VO services and users. The architecture was implemented in the EU FP6 project GridTrust[1]. In this architecture, the VO Management subsystem consists of other services in addition to the *Reputation Management* (RM) service, such as a *VO Manager* (VOM), a *Reputation-aware Service Broker* (RSB) and a *Service Usage Control and Monitoring* (SUCM) service. The system is shown in Figure 1.6.

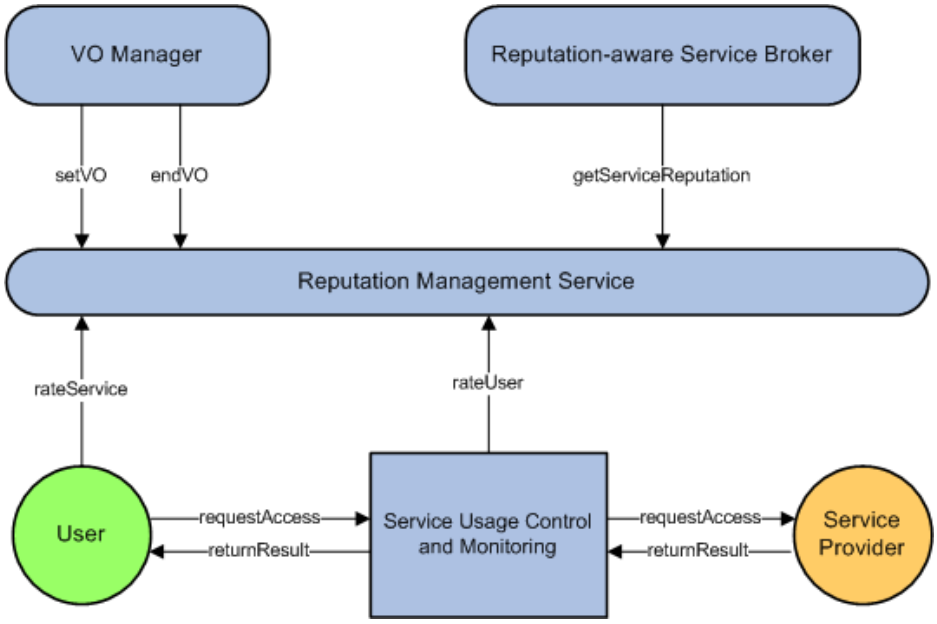


Figure 1.6: Reputation Management in VOs

The VOM informs the RM service of the setting up of a new VO, which

[1]http://www.gridtrust.eu

includes the registration of the list of VO services and users, as shown in the following protocol:

VOM→ RM : setVO(VO ID, Service ID List, User ID List)
RM→ VOM : ack()

Where *VO ID* is the identity of the VO being registered, *Service ID List* is the list of services of the VO, and *User ID List* is the list of users in the VO. On the other hand, the termination of an existing VO is carried out through the following protocol:

VOM→ RM : endVO(VO ID)
RM→ VOM : ack()

Where *VO ID* is the identity of the VO being terminated. The RSB service is used during the setting of new VOs by the VOM. During this phase, the RSB may request from the RM service the reputation of a service in a particular VO or in the general VBE before proposing it to the VOM:

RSB→ RM : getServiceRep(Service ID, VO ID)
RM→ RSB : return(Service ID, Reputation Value)

In case the VO ID is assigned a NULL value, the returned reputation will be the service's reputation in the general VBE.

The SUCM service is a service that monitors requests and replies sent to and from a service in its interaction with a VO user. The SUCM service can detect any undesirable behaviour by the user in its usage of the service being protected by that instance of SUCM. This could be for example the excessive storage of data on resources underlying the service beyond the user's quota. Hence, the SUCM service can report prohibited actions performed by the VO users to the RM service as follows:

SUCM→ RM : reportUser(Service ID, User ID, VO ID, Action)
RM→ SUCM : ack()

The RM service can also accept ratings by the VO users of the QoS levels they have experienced in their interactions with VO services. This is done through the following protocol, in which the user reports the QoS value:

User→ RM : rateService(User ID, Service ID, VO ID, QoS Value)

RM→ User : ack()

Finally, any of the entities in a VO may request from the RM service the reputation of a user:

Any→ RM : getUserRep(User ID, VO ID)

RM→ Any : return(User ID, Reputation Value)

Again, in the event that the VO ID is assigned a NULL value, the returned reputation will be the user's reputation in the general VBE.

1.3.5.1 Usage Scenario

We consider here an example of a usage scenario of the RM system as shown in Figure 1.7. We assume that a RSB starts by querying the RM system for the reputation of a couple of services, Service1 and Service2, in order to join them to a new VO. After that, the VOM signals to the RM system the setting up of the new VO and informs the latter of the two services and three users, User1, User2 and User3. Once this operation is acknowledged by the RM system, the VO becomes operational and the users can avail of the services offered.

At some stage, the SUCM service at Service1 captures a prohibited action performed by User3 and thus reports it to the RM system. Based on the utility function for Service1 and the penalty for the prohibited action, the RM system computes the satisfaction of Service1 regarding this action and updates accordingly the different reputation values for User3. Some time later, the SUCM service for Service1 requests to obtain the new reputation value for User3. Based on this new value, SUCM revises its decision to grant access to User3 to use Service1. This may or may not change the access right for User3. Finally, the VOM decides to end the VO (e.g. as a result of achieving its goals) and informs the RM system

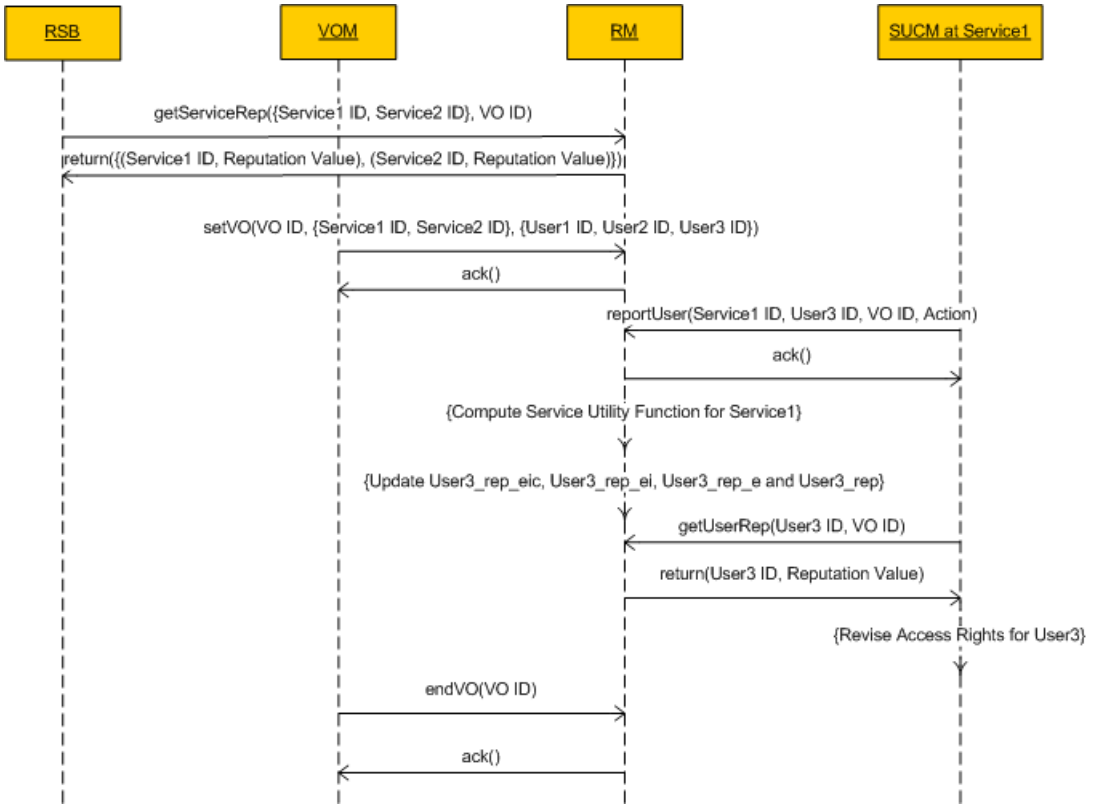


Figure 1.7: Usage Scenario of the Reputation Management System

of this decision. The RM system acknowledges this decision.

1.3.6 Analysis of the Reputation Models

This section describes the results we obtained by performing simulations with various VO setups. We have run our experiments using the SimGrid simulator [Legrand et al., 2003] on which we implemented the following VO operation scenarios:

- VOs with reputation-rated resource providers;
- VOs with reputation-rated users; and
- VOs with reputation-rated resource providers and rated users.

In all simulated scenarios, we compared the results against the case when reputation is not considered to enhance resource management in VOs.

We emphasize from the very beginning on the advantages of our reputation model. First, because we base our model on a-priori collected utility functions, we eliminate one of the biggest drawback of reputation-based trust: the subjectivity of the collected feedback. If the service consumer does not provide truthful information, the provider will be unable to compile the real utility function and will deliver the service accordingly. Thus, as the rational consumers desire to obtain the expected quality of service, they can not manipulate the reputation of a provider without suffering themselves from worsening the received service quality.

Second, our virtual environment is equipped with a trustful reputation manager. Thus, all messages containing valuable information for computing the reputation can not be tampered by malicious nodes. As the reputation manager collects all the individual utility functions, it can unify the perception about the service delivery via the computed reputation measure.

With our reputation model we tackle a major drawback of the quality of service measurement: the fact that various nodes in the system have different representations about which is a good quality delivery.

1.3.6.1 VOs with reputation-rated resource providers

First, we considered a VO with users submitting requests to resource providers for a service. We allow 20% of the providers to produce random QoS values uniformly distributed in a variation band between $85 - 105\%$ of the agreed SLA expected quality. For scheduling the requests to VO nodes we used the *Res_rep_e* value and we allowed that each node will obtain a number of service reuqests proportionaly with its reputation. For a batch of jobs originating from the VO users, we computed the *total completion time* and the *total welfare* produced in the system. Total welfare is obtained by suming all utilities acquired by the users for the submitted jobs. We varied the load factors of the system. The load factor is defined as the proportion of the requested system capacity at a given moment of time vs the total available capacity to be delivered by the VO. For comparison, we allowed the resource broker to schedule the requests in a round-robin fashion. Figures 1.8 and 1.9 show the results.

We should note that with using a reputation-based scheduling, the total completion time is better with around 25% for every load factor of the system. More,

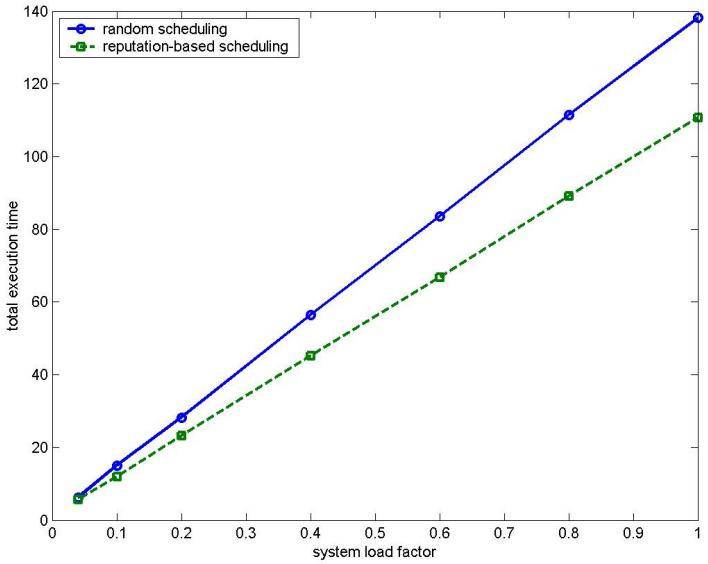


Figure 1.8: Comparing reputation-based scheduling with round-robin: completion time

the system produses 25% much welfare with reputation. The gain in welfare was registered due the fact that the reputation manager has a global view about the quality of the service - through the a-priori collected utility functions. Thus, it can enhance the borkering by selecting appropiate providers for a given consumer.

1.3.6.2 VOs with reputation-rated users

Next, we simulate the system with unreliable users. An unreliable user tries to execute un-permitted actions. We set the fraction of unreliable users to $f = 20\%$, each introducing malicious actions with a sabotage rate of $s = 20\%$. Each action gets a random penalty from $[0, 1)$. Each un-permitted request is identified and refused by the system after its execution and its result gets discarded and never reaches back the user. We should note that the system is now loaded with a fraction fs of malicious requests and it spends some useless time to fulfill those requests.

To simulate this setup, we use a resource centric brokering approach: for an available resource, the next action/job is selected from the available ones

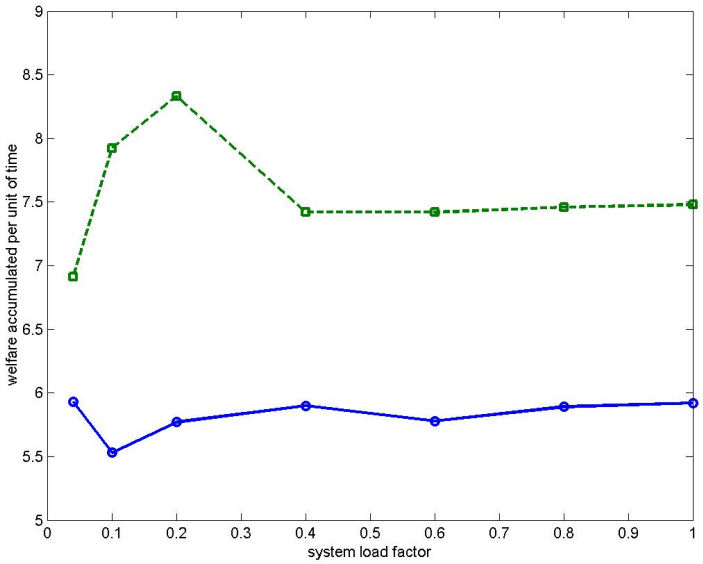


Figure 1.9: Comparing reputation-based scheduling with round-robin: welfare

according with the reputation of the user who entered the action in the system. For reputation, we used the *User_rep_e* value. Such a broker will postpone the execution of an action originating at a less reputed user as late as possible, maximizing the efficiency of the spent running time of the resource. Figure 1.10 shows how the proportion of time spent on processing trustful actions varies during the VO existence. We compared the reputation-enhanced broker with a broker that schedules the tasks on the first-come first-served principle.

We can note that at the beginning of the VO life, the reputation-enhanced broker (depicted with dotted line) schedules only tasks/actions coming from reputed users. The broker starts to schedule tasks from less reputed users only when tasks get scarce in the VO. More, we can note that the overall fraction of time spent on tasks computing is bigger than in the case of a first-come first-served broker. Thus, the reputation manager succeeds to perform a distinction between malicious users and the truthful ones, enabling an enhanced brokering of the reputed users tasks.

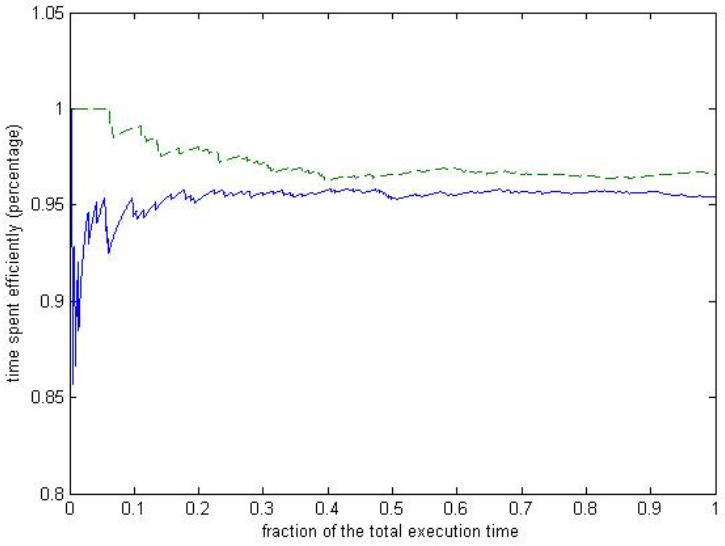


Figure 1.10: Time consumption efficiency for a system with malicious users. We allowed 20% of users to be malicious and having a sabotage rate of 20%.

1.3.6.3 VOs with reputation-rated resource providers and rated users

A more complex scheduling is the one that selects the most reputable available resource and puts on it the job of the most reputable user. This later setup simulates the case of a VO with unreliable users executing actions on unreliable resource providers. Our scheduling intends to protect reputed providers by assigning on them actions from reliable users.

As a benchmark, we used the FIFO (round-robin) scheduling, at each resource executing the oldest request in the system.

Figure 1.11 shows the simulation results. We counted the total welfare (satisfaction) produced and we depicted the welfare acquisition curve for each of the three cases described above. The Y axis plots the proportion of the total satisfaction perceived by the users during the time. We can note that the case when the broker is aware both about the reputation of users and of resources allows for a quicker welfare accumulation. The case when scheduling is done on the basis of first-come first-serve (in both regarding the resources and the user actions) is the worse, letting the users to accumulate satisfaction only latter in time. We should note that by the middle of the simulation, for a total of about 4% (20%x20%)

malicious actions, we get about 10% more satisfaction acquired.

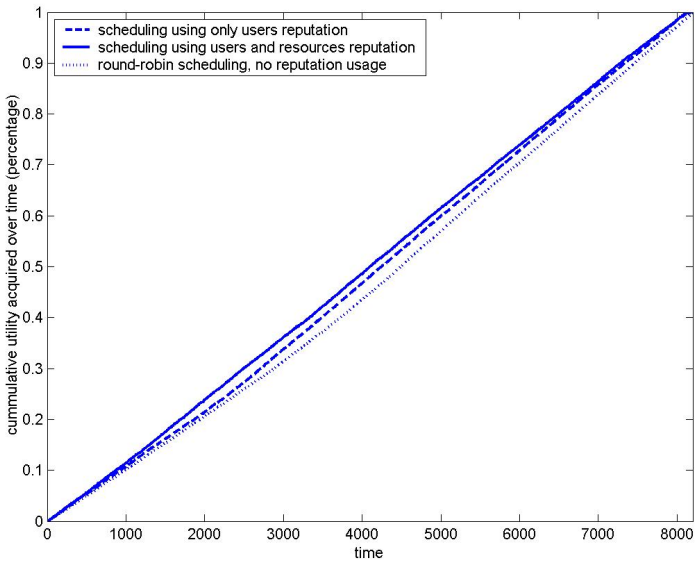


Figure 1.11: Reputation based scheduling with user and resource reputation. We allowed 20% of users to be malicious and having a sabotage rate of 20%.

We should note that in VOs with reliable users and unreliable resources, the reputation-based approach increases the overall system performance in the sense that tasks get faster executed. With unreliable users, using the reputation-based approach the system increases the user satisfaction by allowing trustfull users to benefit first. Furthermore, the reliable resources are used more effective, in the sense that trusted actions are assigned on them.

1.3.7 Conclusion and Future Work

Trust in distributed computing is an important area of research, which contributes to the strengthening of security and the enhancement of the robustness of information sharing. Reputation is one measure by which trust can be quantified and reasoned upon.

Our main contribution in this research direction was to define a model for reputation management in collaborative service-oriented virtual organisations that is based on utility computing and that can be used to rate users according to

their service usage and service providers and their services according to the quality of service they deliver. We also demonstrated, through Grid simulations, the behaviour of the model regarding completion time and welfare, both of which showed improvements over non-reputation-based VOs.

There are several areas in which our model can be extended and improved. The portability of reputation across VBEs is one such area in which an entity or a consumer who change their community (VBE) can bring along their reputation from previous communities to any new ones they join. Another area is to enhance the model to be able to express the reliability of events by adding weights to them. Also, the language of the issues of interest is currently left undefined (apart from the service QoS and service Usage terms). Such a language is domain-specific and it can be obtained from domain-specific ontologies. Finally, an interesting area is to model the reputation of orchestrated services based on the reputation of the individual services. This requires the definition of a *reputation composition* operator.

1.4 Sabotage Tolerance in Volunteer Computing

In this section, we present our scientific contribution regarding models for tackling sabotage in volunteer computing and desktop grids. Initially, we investigated models for sabotage tolerance in Silaghi et al. [2008b]. Further, we described the colluding sabotage behavior and presented a model to tackle nodes collusion [Silaghi et al., 2008a, 2009]. Our approach was recognized by the scientific community as the first to describe the colluding sabotage problem for desktop grids and propose a solution to it [Canon et al., 2010]. We further contributed to define a more sophisticated sabotaging behavior and investigate a solution based on the *maximum independent set* concept [Araujo et al., 2011]. In this section, we will emphasize on the scientific contributions developed in Silaghi et al. [2008a, 2009], and we will shortly brief MIS-based approach of Araujo et al. [2011].

Desktop Grid systems reached a preeminent place among the most powerful computing platforms in the planet. Unfortunately, they are extremely vulnerable to mischief, because computing projects exert no administrative or technical control on volunteers. These can very easily output bad results, due to software or hardware glitches (resulting from over-clocking for instance), to get unfair computational credit, or simply to ruin the project. To mitigate this problem, Desktop Grid servers replicate work units and apply majority voting, typically on 2 or 3 results. In our contribution, we observe that simple majority voting is powerless against malicious volunteers that collude to attack the project. We argue that to identify this type of attack and to spot colluding nodes, each work unit needs *at least* 3 voters. In addition, we propose to post-process the voting pools in two steps. *i*) In the first step, we use a statistical approach to identify nodes that were not colluding, but submitted bad results; *ii*) then, we use a rather simple principle to go after malicious nodes which acted together: they might have won conflicting voting pools against nodes that were not identified in step i. We use simulation to show that our heuristic can be quite effective against colluding nodes, in scenarios where honest nodes form a majority.

1.4.1 Research objective

Internet Desktop Grids [Anderson, 2004; Cappello et al., 2005] aggregate huge distributed resources over the Internet and make them available for running a growing number of applications. In 2007, BOINC [Anderson, 2004], the most popular desktop grid (DG) platform, runs about 40 projects, aggregating more than 400,000 volunteer computers yielding on daily average more than 400 TeraFLOPS [Anderson and McLeod, 2007]. Numbers are even larger by now, BOINC reporting more than 5500 PetaFLOPS, as of start of February 2012.

A major concern in such a middleware is the support for sabotage tolerance (ST). Since computations run in an open and non-trustable environment, it is necessary to protect the integrity of data and validate the computation results. Without a sabotage-detection mechanism, a malicious user can potentially undermine a full computation that may have been executing during weeks or even months [Domingues et al., 2007].

All important ST techniques designed up-to-date for Internet DGs are based on the strong assumption that workers are *independent* from each other. While this assumption is fulfilled, actual sabotage tolerance techniques perform very well, supplying the required (very low) error rate for the overall computation. In particular, BOINC uses replication with majority voting which can bring an error rate of about 1×10^{-5} by validating each result with only two similar responses [Kondo et al., 2007]. But, as Zhao et al. [2005] acknowledge, a potential threat comes up when workers can devise some scheme to interact, for example, with a distributed hash table.

However, there are many signs suggesting that this kind of peer-to-peer interaction among peers will become a standard in the near future. For example, collaborative techniques are very attractive for data distribution [Costa et al., 2008; Wei et al., 2005], especially when the DG runs a parameter sweep application. More, research advance in the direction of an entirely distributed P2P desktop grid [Kim et al., 2007]. BitDew [Fedak et al., 2008] platform is functional, providing services for management and distribution of data on DGs. Currently, it supports the data distribution for the new MapReduce computing paradigm [Tang et al., 2010], emerging on top of the Desktop Grids.

While very powerful and well intentioned, all these solutions bring a side effect

with them: they can help malicious workers to team up to defeat the project, thus violating the workers' independence assumption.

Further, a P2P-enhanced desktop grid might become the target for a Sybil attack [Douceur, 2002]: an individual either creates multiple identities which appear as individual ones to the master or gets control through a virus over a large number of workers. Such a powerful individual can develop a collective malicious behavior if the platform allows for peer-to-peer interaction.

This brings new challenges to the design of a desktop grid system, because the master is not prepared to fight potential collective malicious behaviors, resulting from orchestrated workers.

To face the new collusion threat, we proposed and developed for the first time a novel approach as a complement to the actual replication-based mechanism, which is the most popular ST technique employed in the nowadays middleware. With replication, the master decides about the trustworthiness of a result immediately after having collected all replicas of a work unit. Instead, in our approach the master will postpone the decision moment in the replicated voting pools until it gathers enough information to infer the trustworthiness of the workers. We present a statistical tool to analyze together the voting pools and to infer and classify a worker as being malicious or not. Further, the master can mark a voting pool as being suspicious if a honest worker is losing the decision. On these voting pools, the master can apply further replication in order to conclude about the valid result. We first presented this technique in Silaghi et al. [2008a]. In the journal version [Silaghi et al., 2009], we presented a larger set of experiments, including comparisons with k-means [MacQueen, 1967], one of the most well-know clustering algorithms.

In contrast to other works on ST in DGs [Sarmenta, 2002; Zhao et al., 2005], we evaluate our approach considering a wider range of malicious saboteurs, including naive and colluding ones, as well as transient saboteurs which change their profile during their life.

Our scientific contribution was recognized as the first one to acknowledge the collusion threat for volunteer computing environments [Canon et al., 2010]. Therefore, scientific community came out with other approaches to tackle this problem [Canon et al., 2010, 2011; Staab and Engel, 2009; Watanabe et al., 2009a]. We further contributed to another graph-based approach against much

more subtle colluding attackers [Araujo et al., 2011].

This section is further organized as follows. In subsection 1.4.2 we define the type of saboteurs our approach covers and make a discussion about how efficient the respective sabotage strategies are. In subsection 1.4.3 we present our collusion-resistant sabotage tolerance technique. We start by showing how we can statistically model the voting behavior of workers and how we can classify the workers in malicious and not malicious ones. Next, we present our global sabotage tolerance protocol. In subsection 1.4.4 we present and discuss the results obtained with our sabotage tolerance protocol. In subsection 1.4.5 we shortly brief the maximum independent set approach to tackle the saboteurs that are aware the master employs the approach described in subsection 1.4.3. Finally, subsection 1.4.6 concludes this chapter.

1.4.2 Background

A desktop grid system consists of a server (referred further as the *master*) which distributes work units of an application to *workers*. Workers are machines which voluntarily join the computation over the Internet. Once a work unit is completed at the worker site, the result is returned back to the master. A result error is any result returned by a worker that is not the correct value or within the correct range of values [Kondo et al., 2007].

The *error rate* ϵ is defined as the *ratio* of bad results or errors among the final results accepted at the end of the computation. Thus, for a batch of N work units with error rate ϵ, the master expects to receive ϵN errors. For every application, the master employs some sabotage-tolerance mechanism for obtaining an acceptable error rate ϵ_{acc} with regard to its application.

Redundancy is defined as the ratio of the total number of replicas assigned to workers to the actual number N of work units. Usually, redundancy is larger than 1, which means that we spend computing resources only for verification purposes.

Below, we present the state-of-the-art research regarding sabotage tolerance in desktop grids and volunteer computing

1.4.2.1 State-of-the-art

In volunteer computing, the sabotage tolerance concept was firstly coined by [Sarmenta, 2002], to present techniques that can overcome the presence of malicious volunteers in the contributors set. Early works on sabotage tolerance [Anderson, 2004; Du et al., 2004; Golle and Mironov, 2001; Sarmenta, 2002] assume a basic naive model of saboteurs. These workers make mistakes with some individual sabotage probability, independently of the behaviors of other nodes.

Given these assumptions, majority voting-based replication [Anderson, 2004; Sarmenta, 2002; Taufer et al., 2005] was conceived as a simple but very effective [Kondo et al., 2007] tool to fight saboteurs. The master distributes $2m-1$ replicas of a work unit to workers and when it collects m similar results, it accepts that result as being the correct one. Each collected result is seen as a vote in a voting pool with $2m-1$ voters and with majority agreement being the decision criteria.

For this basic naive behavior, other more complicated techniques were devised, including result verification based on sampling [Du et al., 2004; Golle and Mironov, 2001; Watanabe et al., 2009b; Zhao et al., 2005] and credibility [Sarmenta, 2002; Watanabe et al., 2009b; Zhao et al., 2005]. Sampling means that the trustworthiness of each worker is verified from time to time, e.g. by administrating them some quizzes or by auditing their results. If a worker result is found erroneous, that worker is blacklisted, in the sense that all its previously and future results are discarded. In credibility-based schemes, the master observes the behavior of workers during the computation and estimates their reliability. It assumes that hosts that have computed many results with very few errors are more reliable than hosts with a history of erroneous results. This method has problems to fight against hosts that behave well for a long period of time, in order to gain credibility, and after that start to sabotage.

To be effective, sampling and credibility is often combined with wise scheduling techniques [Choi and Buyya, 2010; Watanabe et al., 2009b].

Kondo et al. [2007] proved the effectiveness of replication in BOINC-like setups, demonstrating that an error rate of 10^{-2} per project can be achieved, given the real-life observed behavior of workers. Replication has the big drawback of the computation cost - each task being scheduled to at least two workers. Sampling and credibility leave the door open for other attacks: a worker can behave

well for a long period of time in order to gain reputation and attack after that.

With only naive saboteurs, Wong [2006] exploits the graph of nodes reflecting how these nodes are linked together in voting pools. This protocol represents a variation of the replication with only 2 replicas and allows the host to estimate, without auditing, the proportion of untrusted workers and how often these workers would submit incorrect results.

For the naive model of saboteurs, a detailed discussion can be found in Domingues et al. [2007]. Choi and Buyya [2010] develop a taxonomy of result verification methods, including voting, sampling and reputation.

We introduced [Silaghi et al., 2009] more sophisticated malicious behaviors in volunteer computing, further detailed in subsection 1.4.2.2. Given that some out-of-band communication is enabled between workers, they can collude and attack in concentration. To detect the colluding behavior, in this chapter we described a statistical tool, which combined with further result audits, removes the negative effects of nodes collusion on the global computation.

The collusion problem was also analyzed by Staab and Engel [2009]. Based on the probability that two nodes are together in the same majority/minority group of a vote, they arrange the nodes in an undirected weighted graph and employ a clustering procedure to partition the graph in a cluster of nodes that correlate highly. This approach is very effective when the correlation probability of two nodes is estimated based on more than 10 observations in average, which might be difficult to attain in practical distributed system setups, characterized by a huge number of nodes and a very low probability that two nodes meet together in more than two voting pools.

Canon et al. [2010] evaluate a setting close to our own. They present two algorithms implementing a dynamic merge/split of groups of nodes with collusion or agreement behavior, given that workers behavior is stationary during their participation in the system. They conclude that the collusion method is slightly better than the agreement one, while the agreement one is much simpler to implement because it does not rely on an external result certification mechanism. To defeat malicious collusion, Canon et al. [2011] develop a combined scheduling and certification algorithm, based on the previously characterization of the workers' behavior.

From the simple master-worker computing model that stays at the foundation

of the desktop grid computing platforms [Anderson, 2004; Cappello et al., 2005], in the last years desktop grids adopted more sophisticated computational models, including MapReduce [Tang et al., 2010]. Within this new computation model, various solutions were considered to handle the saboteurs. We mention here the work of Moca et al. [2011] who analyzed the effectiveness of simple replication with MapReduce; Wei et al. [2009] who designed secure components of a MapReduce desktop grid based on replication and Wang and Wei [2011] who propose a framework towards a secure Hadoop MapReduce with combination of replication and sampling to deal with collusive and non-collusive mappers.

In general, the problem of collusion in voting pools go larger beyond the volunteer computing setup, fitting the Byzantine model [Lamport et al., 1982] of errors of the distributed computing theory. For example, Fernandez et al. [2006] consider sets of binary tasks (with 0 or 1 output) controlled by a master and distributed to workers, similarly as in a desktop grid. They compute upper bounds for the computational effort of volunteers, given some limitations on the power of (Byzantine) malicious nodes. Electronic commerce is another setting that can be modeled with voting pools (games) and is also subject to manipulation. Tsvetovat and Sycara [2000] show that coalitions among customers can be formed to buy many items from the buyer at a lower price and further, to benefit from this. Anonymous environments (like the Internet), with the same players bidding in successive voting pools allow manipulation, which is almost impossible to detect [Yokoo et al., 2005], or voters collusion [Vragov, 2005]. In combinatorial Internet auctions, the concern was to develop anonymity-proof interaction protocols [Yokoo et al., 2005] or false-name proof protocols [Yokoo et al., 2004]. In our distributed computing setup, we rely on the standard master-worker computational model with its simple pull or push scheduling; we differentiate from this huge amount of literature originating in artificial intelligence as we do not design the interaction. We search for the manipulation (sabotage) after it happened and try to limit its effects on the desktop grid system.

1.4.2.2 Sabotaging behaviors

To characterize erroneous hosts, we consider two models that define extreme behaviors: the first behavior is the *naive malicious*, where a node randomly commits mistakes in some work units independently of the behavior of other

nodes. Note that this could possibly happen because the node is faulty, due, for instance, to malfunctioning hardware. In the other extreme, we consider the *colluding nodes* that make their behavior depend from the participation of other malicious nodes in the voting pools. They introduce errors only when they are sure that their sabotage can be successful, for instance, when they know that other malicious nodes are participating in the voting pool, thus forming a majority. While naive malicious nodes expose themselves to be detected and possibly black-listed in a rather easy way, the colluding voters are much more subtle and can easily pass undetected.

We denote the basic naive malicious node by M_1*-type*. A M_1-type worker submits bad results with a constant probability s, called *sabotage rate*. This naive sabotage model assumes that workers are independent of each other and do not communicate. Even if independent workers which do not communicate are very unlikely to submit the same erroneous result, as the sabotage tolerance literature suggests [Sarmenta, 2002], from now on, we assume that all submitted erroneous results are similar, regardless whether the workers can communicate or not.

If we assume the existence of a fraction f of M_1-type saboteurs in the total population of workers, then the expected error rate $\epsilon_{M_1}(f, s, m)$ of the majority-voting replication is given by Equation (1.1) [Sarmenta, 2002]:

$$\epsilon_{M_1}(f, s, m) = \sum_{j=m}^{2m-1} \binom{2m-1}{j} (fs)^j (1-fs)^{2m-1-j} \tag{1.1}$$

Unlike the basic M_1-type, a colluding saboteur (further referred as M_2*-type*) has the will and the means to reach other saboteurs in order to develop malicious coalitions. In model M_2, a dishonest worker w will sabotage only if it finds enough dishonest peers to join it to defeat the honest nodes involved in the same voting pool. Thus, a M_2-type malicious worker will never sabotage without winning the decision in its voting pool. We assume that there is a complete graph connecting all the malicious nodes, such that communication between any two malicious nodes is always possible at any point in time. However, at this stage of our work, we impose a limit to the power of malicious nodes: they are not aware of our sabotage detection mechanisms and they act to conceal themselves from majority voting. With this assumption, colluding saboteurs will attack whenever

they are sure they can win the voting decision against honest workers.

If the fraction of M_2-type saboteurs in the total population of workers is f, each saboteur being an active one with probability s (i.e. s is the probability of a colluding saboteur to launch the coalition-formation protocol), each of them knowing all the rest of the workers, then the expected error rate is given by Equation (1.2).

$$\epsilon_{M_2}(f, s, m) = \sum_{j=m}^{2m-1} \binom{2m-1}{j} \left(1 - (1-s)^j\right) f^j (1-f)^{2m-1-j} \tag{1.2}$$

In eq. (1.2) we have to mention that *at least one* of the j M_2-type workers in a voting pool is sufficient to start (with probability s) the collusion formation protocol, the remaining colluding saboteurs in the voting pool participating by default to the coalition. This leads[1] to the multiplication term $1-(1-s)^j$ before f^j, which denotes the probability of finding exactly j M_2-type workers in a voting pool of size $2m-1$.

We consider yet another type of saboteurs deemed M_3-*type*, mixed malicious, which change their behavior during their life, behaving either naive or colluding, but always performing a dishonest role. For an M_3-type saboteur, we denote with c the *naive ratio*, which is the fraction of work units for which the worker behaves as an M_1-type saboteur with sabotage rate s_1, while for the remaining $1-c$ fraction of work units it behaves like a M_2-type saboteur with sabotage rate s_2. In this case the expected error rate is the one of Equation (1.3):

$$\epsilon_{M_3}(f, c, s_1, s_2, m) = c\epsilon_{M_1}(f, s_1, m) + (1-c)\epsilon_{M_2}(f, s_2, m) \tag{1.3}$$

Given that M_1-type saboteurs submit a rather small fraction s of bad results (with an average of 0.0034 for independent I/O errors [Kondo et al., 2007]), it results that colluding saboteurs are much more destructive than independent ones. Figure 1.12 shows the comparison of the error rates achieved with different number of identical results required m, for $f = 0.035$ and $s = 0.0335$ in the

[1] the overall probability that at least one worker out of j starts the collusion formation protocol, given the individual probability s is: $\binom{j}{1}s(1-s)^{j-1} + \binom{j}{2}s^2(1-s)^{j-2} + ...\binom{j}{j}s^j = 1-(1-s)^j$

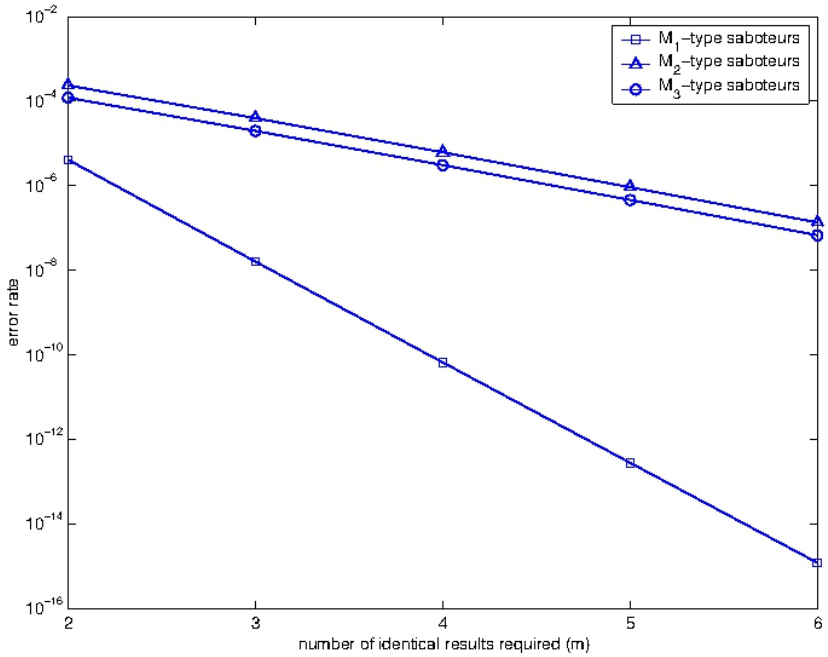


Figure 1.12: Error rates comparison between various types of malicious workers against simple replication

case of both M_1-type and M_2-type saboteurs[1]. To allow for a better comparison, we used the same value of $s = 0.0335$ for all types of colluders in Figure 1.12. However, colluding saboteurs would be much more destructive if they always try to sabotage, i.e., if $s = 1$ (naturally, this can leave more traces of their intervention). The error rate of M_3-type saboteurs is something in between M_1 and M_2-types, being much closer to the latter. We considered $c = 0.5$ for an M_3-type saboteur, while keeping the same f and $s_1 = s_2 = s$.

We define the *effectiveness* of a saboteur as being the ratio between the number of times it succeeds to defeat the sabotage tolerance mechanism versus the total number of times it sabotages. While naive saboteurs succeed to defeat the master's replication-based sabotage tolerance mechanisms only in a small fraction of the attempts, a colluding saboteur will sabotage only when it is sure to win the majority voting, and therefore, its effectiveness is total (1). Of course, the effectiveness of a naive saboteur increases with its sabotage rate s; this being the

[1]We assumed the same error rate parameters as for the top 10% erroneous hosts reported by Kondo et al. [2007].

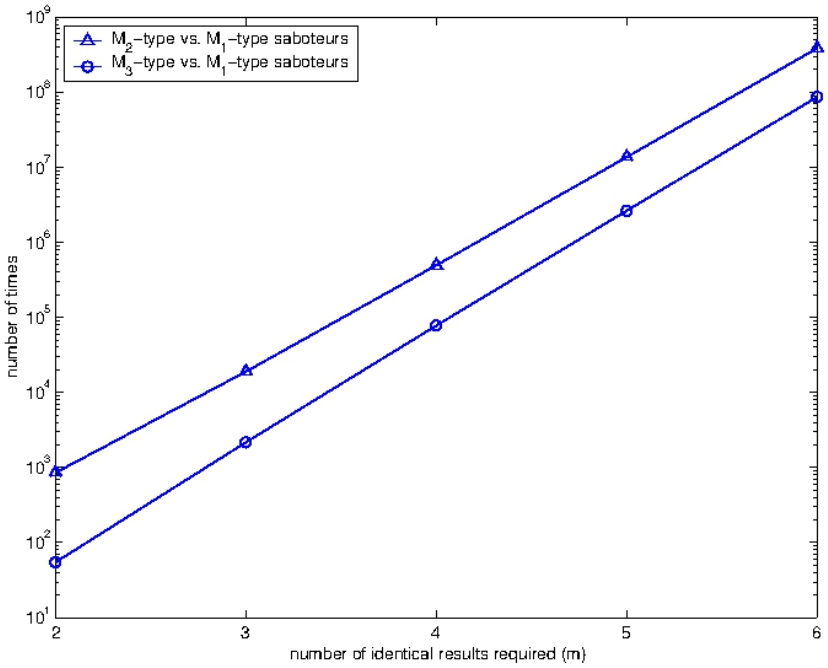


Figure 1.13: Relative effectiveness comparison between various types of malicious workers against simple replication

sole parameter such a saboteur can control. Figure 1.13 depicts the number of times colluding saboteurs of type M_2 and M_3 are more effective than the naive ones for various number of results required. The effectiveness of the saboteurs was computed for the same parameters as in figure 1.12. The effectiveness of the M_2-type saboteurs was computed assuming that they sabotage only when at least a coalition of size 2 is formed. The relative effectiveness of the colluding saboteurs increases exponentially with the number of results required, because the success rate of naive saboteurs is very small in the presence of higher-order replication.

We should also note that besides being less destructive, naive saboteurs leave more traces behind them, making it much easier for the master to spot them out.

1.4.3 A collusion-resistant sabotage tolerance protocol

In this section we propose a collusion-resistant sabotage tolerance protocol. Since the actual replication works very well in the presence of M_1-type naive saboteurs,

we do not intend to replace it. Instead, we complement it with a scheme targeted to spot and defeat colluding saboteurs, which are much more effective and can defeat the replication, as we have seen in subsection 1.4.2.2.

1.4.3.1 Overview

Kondo et al. [2007] demonstrated that replication with $m = 2$ is enough to cope with erroneous hosts with M_1 saboteurs. Additionally, we observe that DG projects that we are aware of, set m to be 2 at most, while some of them initially use only two replicas and ask for another one in case of conflicting responses. Therefore, in our work we fix $m = 2$. This means that the master replicates each task $2m - 1 = 3$ times. However, instead of deciding on a result as soon as the master gets a majority of 2 similar responses, it will wait and postpone the decision until it gets all three results from that work unit and until it collects enough results of related workers from *different* work units. We further consider each work unit as a voting pool, where each worker is worth a vote. After it collects a number of voting information (the most it collects the better), the master will analyze the information acquired from the voting behavior and will infer which are the M_1-type naive saboteurs. The rationale for this is that, once these nodes are identified, the remaining contradictory voting pools only contain colluding nodes of type M_2 and M_3. Then, the master will reconsider these work units and ask for further responses.

The master's objective is to spot out malicious workers, regardless of the sabotage model they fit in. From this point of view, a voting pool that contains contradicting votes is of interest for the master, because it contains, at least one faulty node. If the size of the voting pool is 3, this means that one loosing worker voted against two opponents. A valuable observation is that in the case of naive M_1-type workers, the total number of such conflicting voting pools is higher than in the case of M_2 saboteurs, for the same f and s, regardless their value. This makes it easier for the master to spot out naive saboteurs than colluding ones. While a hybrid M_3 saboteur has a mixed behavior switching between being naive and colluding, this model will give us less clues than naive saboteurs, but more clues than with colluding ones. Therefore, given that M_2 and M_3 nodes are not aware of our sabotage tolerance mechanism and only try to defeat majority voting, we expect a better response against M_3 than against M_2 saboteurs. Figure

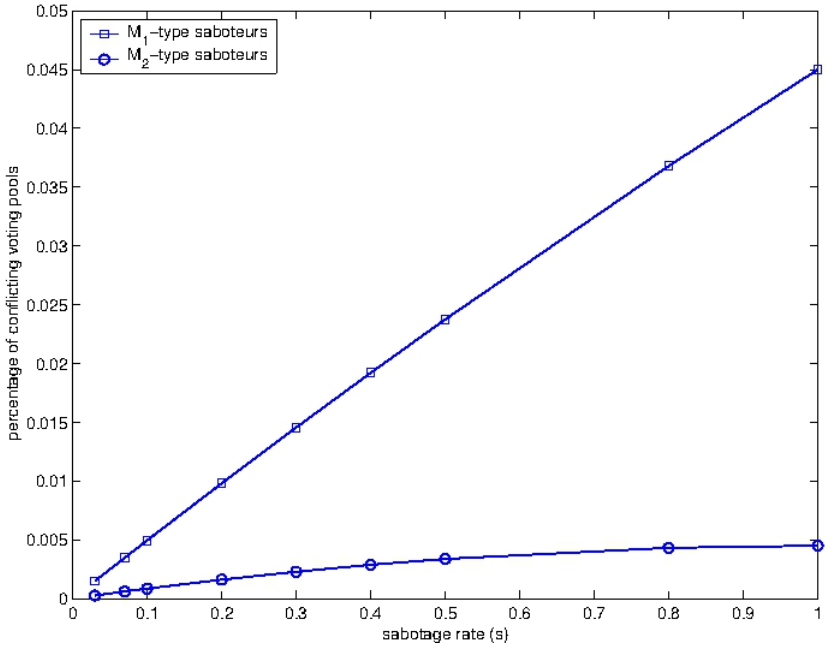


Figure 1.14: Comparison of conflicting voting pools

1.14 shows the percentage of conflicting voting pools for various s, given $f = 0.1$, in pure populations with only M_1-type (respective M_2-type) saboteurs.

Considering a conflicting voting pool, it would be of interest for the master to assess if the worker that is losing the decision is behaving like a naive M_1-type saboteur or not. The master can do this assessment if it possesses a theoretical model of the voting pools world and a classification tool, which we will describe in the following section.

1.4.3.2 Statistical modeling of the voting behavior

Consider a population $\mathcal{S}_P$ consisting of honest and malicious workers. Table 1.2 describes the meaning of each structure parameter. We impose that honest workers are in majority, i.e., $f_1+f_2+f_3 < 0.5$. To enable evaluation, we assume that the population structure is stable over time and the workers fully comply with their models during all their life. Additionally, there is an implicit assumption in the models of workers that we devised: nodes are unaware of the algorithm used by the master to spot collusion. Let the master distribute replicated tasks from a set of work units $\mathcal{S}_W$, such that, on average every worker gets N tasks.

Table 1.2: Parameters describing the population structure

f_1	proportion of M_1-type workers
s_1	sabotage rate of M_1-type workers
f_2	proportion of M_2-type workers
s_2	sabotage rate of M_2-type workers
f_3	proportion of M_3-type workers
c	naive ratio for M_3-type workers
$s_{3,1}$	sabotage rate of M_3-type workers while behaving as M_1-type
$s_{3,2}$	sabotage rate of M_3-type workers while behaving as M_2-type

A voting pool $V = \{v_1, v_2, v_3\}$ is a set of three ($m = 2$) different workers $v_i \in \mathcal{S}_P$, each of which submitting a binary vote in the pool. Consider a fixed worker $v \in V$. The number of *votes against* the worker collected in the voting pool V can be modeled as a random variable $Y_v : \{0, 1, 2\} \to \mathbb{R}$, where $Y_v(i) = p_{v,i} \geq 0$ is the probability that the worker v has i votes against in the voting pool V, with $\sum_i p_{v,i} = 1$.

Due to the *i*) population structure stability; *ii*) worker's fully compliance with its model; and *iii*) the fact that workers can not influence how the master distributes them in the voting pools, any two voting pools for the same worker are statistically identical and independent and thus, we can model the behavior of a worker during a sequence of N voting pools as a multinomial experiment with N trials Y_v.

We denote by $Y_{v,N} : \{0, 1, \ldots, 2N\} \to \mathbb{R}$ the random variable defining the probabilities for the worker v to collect a given number of votes against over a total of N voting pools.

From the independence between two different voting pools, we can infer that $Y_{v,N} = \prod_{t=1}^{N} Y_v = Y_v^N$. [1] In our case, as every Y_v is defined over the set $\{0, 1, 2\}$, for the sake of simplicity, the discrete values of the random variable $Y_{v,N}$ can be obtained by computing the corresponding coefficients of a polynomial like the

[1] Given 2 random variables $Y_1 : \{x_i,\ i = 1, \ldots, n_1\} \to \mathbb{R}_+,\ Y_1(x_i) = p_i,\ \sum_i p_i = 1$ and $Y_2 : \{x'_i,\ i = 1, \ldots, n_2\} \to \mathbb{R}_+,\ Y_2(x'_i) = p'_i,\ \sum_i p'_i = 1$, the product $Y = Y_1 Y_2$ is defined by over the space $\{x_i \wedge x'_j,\ i = 1, \ldots, n_1,\ j = 1, \ldots, n_2\}$ with the following expression: $Y(x_i \wedge x'_j) = p_i p'_j$.

one of Equation (1.4):

$$(p_{v,0} + p_{v,1}X + p_{v,2}X^2)^N \tag{1.4}$$

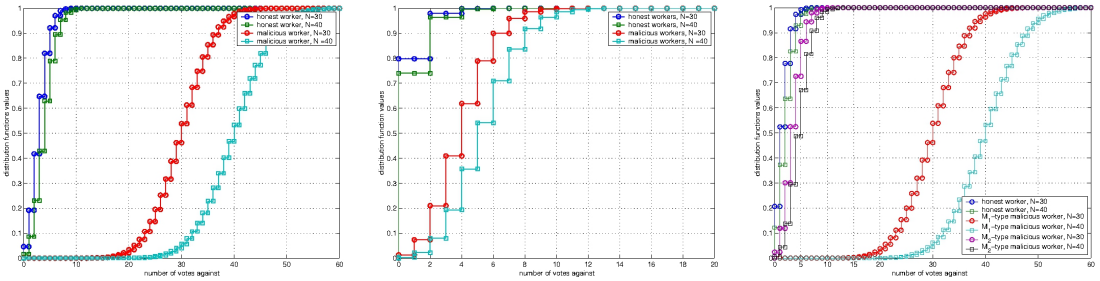
These coefficients can be computed either by successively multiplying the polynomials (as we did) or by applying the multinomial theorem and using the trinomial coefficients [Karlin and Taylor, 1975].

As an example, if the random variable of a worker v is $Y_v = \{0.6, 0.2, 0.2\}$, meaning that the worker scored 0 votes against in 60% of cases, 1 votes against in 20% of cases and 2 votes against in 20% of cases, if the worker participated in 5 voting pools, then random variable $Y_{v,5}$ can be obtained by computing the polynomial $(0.6 + 0.2X + 0.2X^2)^5$: which gives $0.0778 + 0.1296X + 0.216X^2 + 0.2016X^3 + 0.1776X^4 + 0.1059X^5 + 0.0592X^6 + 0.0224X^7 + 0.008X^8 + 0.0016X^9 + 0.0003X^{10}$. The random variable $Y_{v,5}$ consists of the coefficients of the before-computed polynomial and should be read as follows (e.g.): the probability that after 5 voting pools the worker v registers 6 votes against is 5.92%.

The *joint distribution function* of a voter v with $Y_{v,N}$ is $F_v : \{0, 1, \ldots 2N\} \rightarrow \mathbb{R}$, defined as $F_v(i) = Prob(Y_{v,N} \leq i)$, $F_v(i)$ being the summation of all coefficients of the polynomial (1.4) up to the i rank.

In the heart of our heuristic lies a simple intuition about the distribution functions F_v: if we compare F_v for a honest node and for an M_1 malicious node there is a huge separation between both lines, because a typical honest node gets much fewer votes against than a typical M_1 node. For a given population structure, after determining the initial values $p_{v,0}$, $p_{v,1}$ and $p_{v,2}$ and computing the coefficients of Equation (1.4) using multiplications of polynomials, we got distribution function curves like the ones depicted in Figure 1.15. The population we used to plot these curves was the following: in each of them we considered $f = 0.1$ malicious workers. Each worker has some predefined sabotage rate of 0.5 and we assigned once $N = 30$ and $N = 40$ work units per worker. First, in Figure 1.15(a) we considered only naive M_1-type workers. In Figure 1.15(b) we replaced naive M_1-type workers with colluding M_2-type workers. We can notice that for the same percentage of the malicious workers ($f = 0.1$) and the same sabotage rate ($s = 0.5$), the gap between the distribution functions for $N = 30$ and $N = 40$ increases, while the distribution function of naive workers shifts to the right. In Figure 1.15(c) we considered a mix of M_1 and M_2-type workers, keeping the proportion of malicious workers identical ($f_1 + f_2 = 0.1$). We

can notice that the distribution function of the naive malicious is on the right side, the distribution function of the honest workers is on the left side, while the distribution function of the colluding malicious is shifted a bit on the right of the honest workers distribution.



(a) Honest and M_1-type workers

(b) Honest and M_2-type workers

(c) Honest, M_1 and M_2-type workers

Figure 1.15: Theoretical distribution functions F_v for various population structures

After we analyzed extensively various population structures, using the mathematical procedure explained in the above paragraph, the following important conclusions can be drawn out:

- M_1-type (naive) saboteurs always collect the biggest number of votes against, their joint distribution functions being the most-right ones in the graphic;
- for N large enough, there is a clear separation between the distribution functions F_v for the case of honest workers versus malicious workers;
- the honest workers have the distribution functions on the left side of the graphic, the distances between a honest worker distribution and a naive (M_1) malicious one being the biggest ones;
- as expected, the distribution function for an M_3-type worker, not shown on the plots due to space consideration, will lay down between distribution functions of M_1 and M_2 workers, being placed on the left side of the distribution for the M_1-type workers.

1.4.3.3 Spotting out naive saboteurs (M_1 or M_3)

Based on the theoretical conclusions drawn out in Section 1.4.3.2, we now propose a method for spotting out saboteurs that behave permanently or intermittently as naive M_1-type ones. This includes M_3-type workers.

Suppose that the master distributes a batch of work units, such that each worker takes place in an average of N voting pools. For some particular worker i, the number of voting pools is N_i and the master can count the number of times c_0, c_1, c_2, the worker registered 0, 1, and 2 votes against, among its work units. These figures, divided by N_i give the practical (sampled) probabilities p_0, p_1, p_2 (as used in Equation 1.4) for that worker. Applying the procedure described in Section 1.4.3.2, the master will obtain one distribution function (similar to the ones of Figure 1.15) for each worker.

Figure 1.16 depicts the distribution functions for workers participating in an experiment with a population structure with $f_1 = f_2 = 0.1$, $s_1 = s_2 = 0.5$ and $f_3 = 0$, for $N = 30$ and for a small number of nodes (in order to facilitate the display of the distributions on the plot). As expected from the theoretical results presented above, the distribution functions of M_1-type workers (solid lines) will agglomerate the right side of the plot, while the ones of the honest workers (circled lines) and M_2-type workers (squared lines) will stay on the centre and left side. The honest workers that were not placed in voting pools with malicious opponents have the most left-sided distributions.

For two voters $v_i \neq v_j$ with the distribution functions F_{v_i} and F_{v_j} computed after considering all voting pools they took place in, we define in Equation 1.5 the distance between their distribution functions:

$$d(v_i, v_j) = \sum_k (F_{v_i}(k) - F_{v_j}(k))^2 \tag{1.5}$$

Now, consider the symmetrical matrix $D = (d_{i,j})$ of size $n \times n$, where its elements are defined as the distances $d_{i,j} = d(v_i, v_j)$. A row i of this matrix shows how statistically different is the behavior of worker v_i from the rest of workers in the population. The matrix D can be normalized to a matrix C to make the values of each row sum 1, by dividing each row by its own sum.

According to the theoretical findings (Section 1.4.3.2), the distances between naive-behaving saboteurs and the majority of the population should be large.

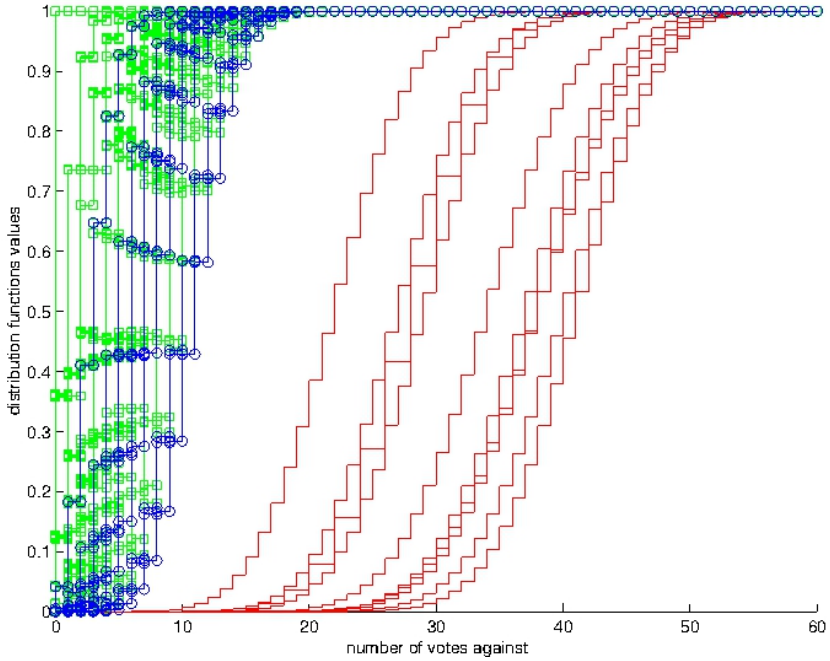


Figure 1.16: Distribution functions for a population with M_1 and M_2-type saboteurs

Having in matrix C a measure of distance between any pair of nodes, we can use the EigenTrust algorithm of Kamvar et al. [2003] (Algorithm 1), to give each node a single global score (its corresponding eigenvalue).

EigenTrust algorithm aggregates private reputation values of a node for other nodes in the network in order to supply with the global reputation value for each node. Given that two nodes are assigned with similar private reputation values from the rest of the nodes in the network, EigenTrust produces global values closed each to another for those two nodes. In our case, the score produced by EigenTrust for each node tells us how likely is that node to be dishonest. Kamvar *et al.* proved that the algorithm will converge to some global scores vector, $\vec{t}$, if the initial matrix C is not singular. More, the global vector $\vec{t}$ contains only positive values with $\sum t_i = 1$.

To avoid obtaining singular matrices, we remove from C the rows and columns for workers that scored only 0 votes against in all their voting pools. After we compute $\vec{t}$, we sort the scores of the nodes in ascending order assuming that each value represents a discrete probability and we compute their corresponding

Algorithm 1: The simple EigenTrust algorithm [Kamvar et al., 2003]

Input data:

$C = (c_{i,j})$ a matrix of size $n \times n$, with $\sum_j c_{i,j} = 1$

some small error ϵ

$\vec{t}^0 = (t_i^{(0)})$, with $t_i^{(0)} = \frac{1}{n}$, for every $1 \leq i \leq n$

repeat

$\vec{t}^{(k+1)} \leftarrow C^T \vec{t}^{(k)}$

$\delta \leftarrow \|\vec{t}^{(k+1)} - \vec{t}^{(k)}\|$

until $\delta < \epsilon$

distribution function.

In Figure 1.17 we depict a particular case for this distribution function (for a population of 1000 workers processing on average $N = 30$ work units each, with $f_1 = f_2 = 0.1$, $s_1 = s_2 = 0.5$). The distribution function registers less than 1000 t-values, because we removed from the population the workers that scored 0 votes against in all their voting pools (around 50 workers in our case).

In most of our experiments we got a clear "knee" (indicated by the arrow) in this plot, resulting from the differences between naive saboteurs and remaining population. Identifying this knee, we can classify the workers in naive malicious and not naive malicious ones. In the absence of a knee, our algorithm should just not mark any worker as naive malicious and assume that all of them are either honest or M_2-type nodes (in fact some of the nodes could be M_3 behaving mostly as M_2).

To locate the knee, when it exists, we use the second order differences[1] of the vector $\vec{t}$ values, as these emphasize in a clearer way the fast growth in the zone of the knee. We consider 10 consecutive values in the second differences and we compute their statistical variance. The knee shows up when these variances go above a given threshold (see Equation (1.6)). In our experiments we set $\vartheta = 10^{-8}$. We also tried other thresholds with a difference of up to 3 orders of magnitude and we noticed no sensible difference. Thus, we can consider $\theta = t_{i_{cr}}$, where i_{cr}

[1]Given the vector $X = \{x_1, x_2, ...x_n\}$, the first order difference vector ∇X is defined as the vector $\nabla X = Y = \{y_1, ..., y_{n-1}\}$ with $y_i = x_{i+1} - x_i$. The second order difference vector $\nabla^2 X$ is defined as $\nabla^2 X = \nabla Y$.

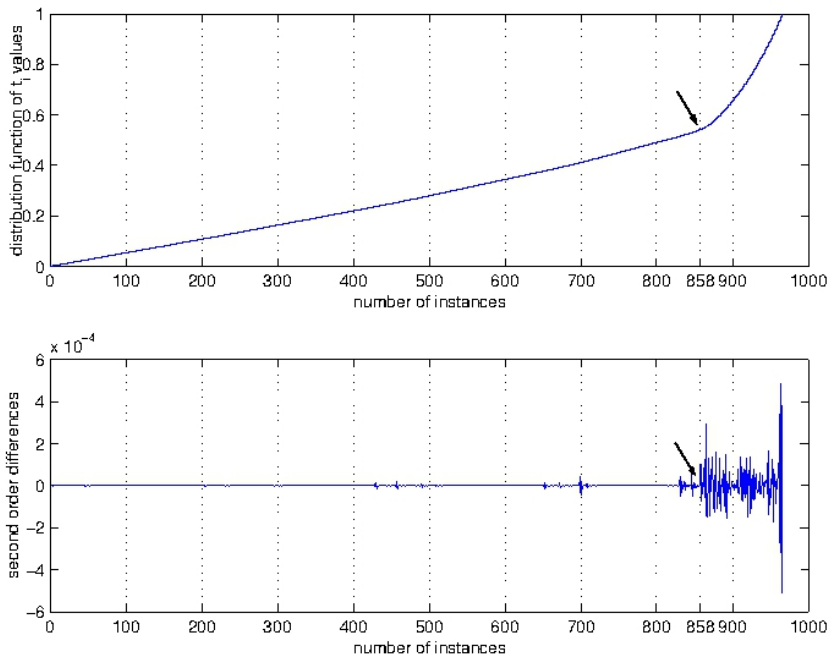


Figure 1.17: Distribution function and second order differences for the $\vec{t}$ values

is given by eq. (1.6) and classify as naive malicious all workers i such us $t_i \geq \theta$.

$$i_{cr} = \max\{i \mid var(\overline{t_{i-10}, t_i}) < \vartheta\} \tag{1.6}$$

In section 1.4.4 we will present the classification results. We should be aware that our final goal is the identification of the colluding saboteurs, while identifying with high certainty the naive saboteurs represents only an intermediary step.

1.4.3.4 A general sabotage tolerance protocol

The theoretical modeling using multinomial experiments presented in Section 1.4.3.2 allowed us to define the classification procedure presented in Section 1.4.3.3. With high certainty, the master can identify malicious workers, especially those of type M_1, while keeping the classification error low - as we will see in section 1.4.4.1. A low classification error means that a small number of false positive workers, which are in fact honest workers, are reported by the classification scheme. In this section we go further and define our general sabotage tolerance protocol.

Because actual replication is effective to defeat naive saboteurs, our protocol

identifies those cases where a worker that is not classified as (naive) malicious is defeated, and asks further replication on those voting pools. Therefore, we do not replace the actual replication-based sabotage tolerance protocol; rather we complement it with a tool to spot out situations when colluding saboteurs win against honest ones. Specifically, the master has to employ the general algorithm described in Algorithm 2.

Algorithm 2: The general sabotage tolerance algorithm

1: **Input data:**
2: $\mathcal{S}_W$: the set of work units, $\mathcal{S}_P$: the set of workers
3: **Begin**
4: $\mathcal{S}_V \leftarrow$ *Distribute_tasks*($\mathcal{S}_W$, $\mathcal{S}_P$, 3);
5: $\mathcal{S}_{V,conflicting} \leftarrow$ Select conflicting pools from $\mathcal{S}_V$
6: $\mathcal{S}_{Mal} \leftarrow$ Identify malicious workers
7: $\mathcal{S}_{V,suspect} \leftarrow$ Select suspect pools from $\mathcal{S}_{V,conflicting}$
8: $\mathcal{S}_{V,err} \leftarrow$ Ask 2 more responses on pools from $\mathcal{S}_{V,suspect}$
9: $\mathcal{S}_{M_2 \cup M_3} \leftarrow$ Identify colluding workers from $\mathcal{S}_{V,err}$
10: $\mathcal{S}_{V,suspect1} \leftarrow$ Identify voting pools with consensus of only colluding workers
11: Ask a new voting pool on every $V \in \mathcal{S}_{V,suspect1}$
12: Accept the results for all $V \in \mathcal{S}_V$ by majority voting
13: **End**

Up to line 4 in Algorithm 2 the master applies the classical replication. In line 5 the master selects the conflicting voting pools out of the initial replication results. Next, in line 6, the master applies the classification algorithm of Section 1.4.3.3 and obtains a list of malicious workers. In line 7, the master selects among the conflicting voting pools those where another worker not a malicious one is defeated. On each of these suspect voting pools, the master ask at most two new response replicas (line 8), by putting the tasks on honest workers. The honest workers are selected from the ones that recorded zero votes against or from the ones that registered the smallest $\vec{t}$ values in the classification procedure. At the end of this step, the master identifies those voting pools $\mathcal{S}_{V,err}$ where the initial result was reverted. From these voting pools, the master identifies the colluding workers (line 9). Next, in step 10, the master traces back all non-conflicting voting pools where three malicious workers where initially assigned.

On each of these voting pools, the master invalidates the initial quorum and asks a new 3-times replication with honest workers as above. In the end, the master accepts the results of each voting pool with a majority voting.

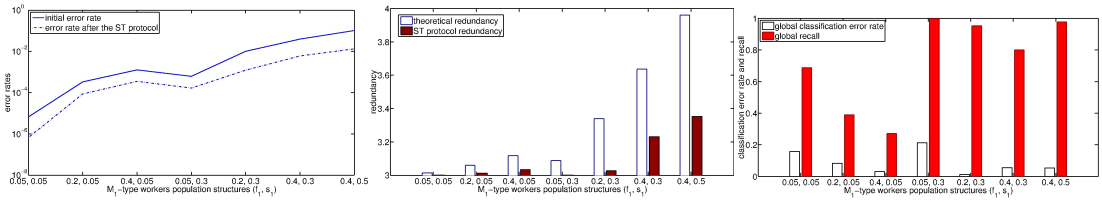
1.4.4 Results and discussion

1.4.4.1 Results

In this section we report the results achieved with our sabotage tolerance scheme. We considered various population structures and we let the master assign tasks such that each worker gets on average $N = 30$ tasks (i.e. 10000 work units for a population of 1000 workers). For each population structure we have run 100 experiments, to get a statistical confidence on our results. However, for convenience, in our plots we show only the average values.

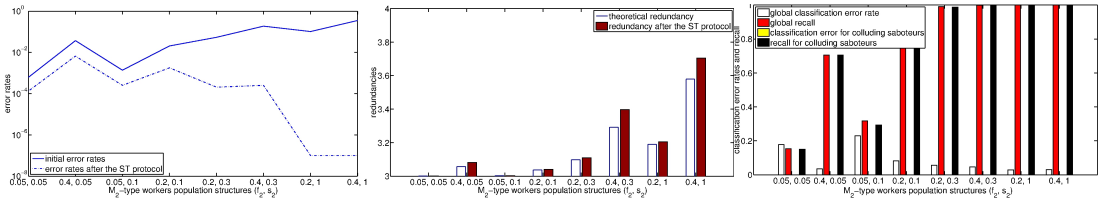
To evaluate the performance of the proposed sabotage tolerance scheme, we computed the final error rate and redundancy obtained with our scheme and we compared them against the ones obtained with the simple replication, before applying our sabotage tolerance protocol. For a cost estimation, we also compared the actual redundancy of our scheme against a "theoretical" redundancy that would be obtained if the sabotage tolerance protocol would ask for another task replica on each voting pool with conflicting responses. This theoretical redundancy is an optimistic value because it is still not enough for establishing the correct result of a conflicting voting pool.

But, we are also interested in discovering the colluding saboteurs, i.e. the ones that get defeated after applying the algorithm presented in section 1.4.3.4. We can see this task as a classification one and evaluate its performance by computing the classification error and the recall rates. As advised in the data mining field [Witten and Frank, 2005], the *classification error* represents the percentage of incorrectly classified examples (false positives) out of the total retrieved ones for some class. The *recall* represents the percentage of the examples of some class identified by the automated classification procedure. We should note that obtaining a low classification rate is usually achieved with a cost of a low recall. In what follows, we will also report the classification error rates and recalls for the classification tasks performed by our sabotage tolerance protocol. More precisely, we will report the global results (i.e. classification error rate and recall with re-



(a) Error rates **(b)** Redundancies **(c)** Classification error rates and recalls

Figure 1.18: Results obtained for a population structure with only honest and M_1-type nodes



(a) Error rates **(b)** Redundancies **(c)** Classification error rates and recalls

Figure 1.19: Results obtained for a population structure with only honest and M_2-type nodes

spect to all saboteurs, regardless of their profile) and the results concerning only the colluding saboteurs (i.e. saboteurs belonging to M_2 and M_3 types).

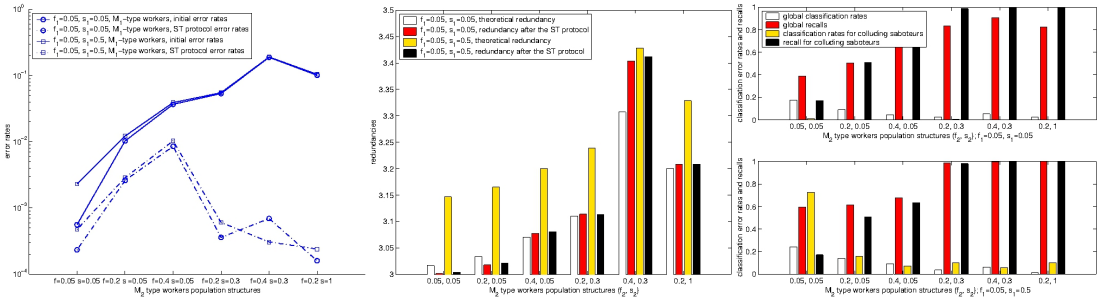
First, we applied our method in pure populations with only M_1 or M_2-type workers (Figures 1.18 and 1.19). We plotted the actual ST protocol error rates (with dotted lines) against the ones obtained with simple replication (with solid lines). We can note in Figure 1.18a that with only M_1-type naive workers, our sabotage tolerance protocol works pretty well, increasing the effectiveness of the replication by at least 10 times and avoiding the verification of each conflicting voting pool (Figure 1.18b). With only M_2-type workers (Figure 1.19a), the sabotage tolerance protocol works in its full power if the workers are sabotaging with rates greater than 0.3, i.e., $s_2 \geq 0.3$. For smaller values of s_2, like 0.05, results of our algorithm are not so good, but even in this case, when simple replication is effective, our protocol succeeds to improve error rate by about 10 times. We can note that defeating all colluding saboteurs (the cases with big sabotage rates)

is done with the cost of a bigger redundancy (Figure 1.19b), as for every conflicting voting pool we ask two new results. This is the reason why in this case the redundancy is higher than the theoretical redundancy. But, we should note that redundancy is still lower than 4 and the percentage of the saboteurs in the population is very high.

Regarding the classification tasks for the M_1-type pure populations (figure 1.18c), our protocol correctly retrieves almost all naive saboteurs with an acceptable low recall, if they sabotage consistently (i.e. $s_1 > 0.3$). Also, the recall is low if there are a considerable number of saboteurs. The worse results are obtained for the cases with $f_1 = 0.05$ or $s_1 = 0.05$, i.e. there are very few saboteurs or they do not reveal out their profile. In this case, classification rates are bigger (for example in the case $f_1 = 0.05$ or $s_1 = 0.3$) because the classification procedure is a statistical one and records some errors by classifying honest workers as M_1-type malicious ones. But, this situation does not affect globally our ST protocol because it introduces only very few additional auditing and redundancy. We can see in figure 1.18a that also in this case, a 10-times gain in ST protocol error rate is obtained. We should note that this population setup evaluates the performance of the initial classification procedure described in section 1.4.3.3, which spots out (with a good certainty) the naive saboteurs.

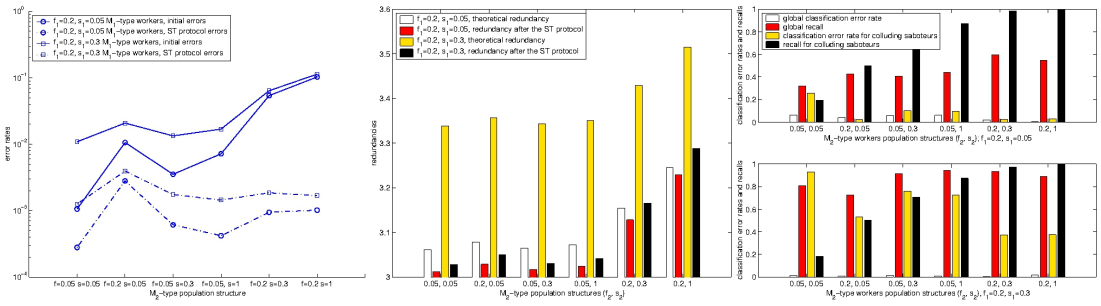
Regarding the classification tasks for the M_2-type pure populations (figure 1.19c), we should note that our algorithm recalls almost all saboteurs if they are in an acceptable proportion ($f_2 > 0.05$) and they reveal their identity ($s_2 > 0.3$). The bigger global classification error rates for the cases when $f_2 = 0.05$ are associated with low recall. This means that step 6 of the algorithm 2 spots out very few naive saboteurs, letting (as expected) the forthcoming replication on suspect voting pools to spot out colluding workers with the price of the increased redundancy. We should also note that the classification error rate for colluding saboteurs is always 0 (figure 1.19c), because in M_2-type pure populations there are no other saboteurs to be classified as colluding malicious.

Next, we considered mixed population structures with naive and colluding workers against honest ones (figures 1.20, 1.21 and 1.22). Specifically, we considered that the naive workers are in a small, medium or large proportions ($f_1 = 0.05$, $f_1 = 0.2$ and $f_1 = 0.4$) and we varied the structure parameters regarding the colluding workers.



(a) Error rates **(b)** Redundancies **(c)** Classification error rates and recalls

Figure 1.20: Results obtained for a population structure with honest, M_1 and M_2-type nodes, M_1-type workers being in a small ($f_1 = 0.05$) proportion

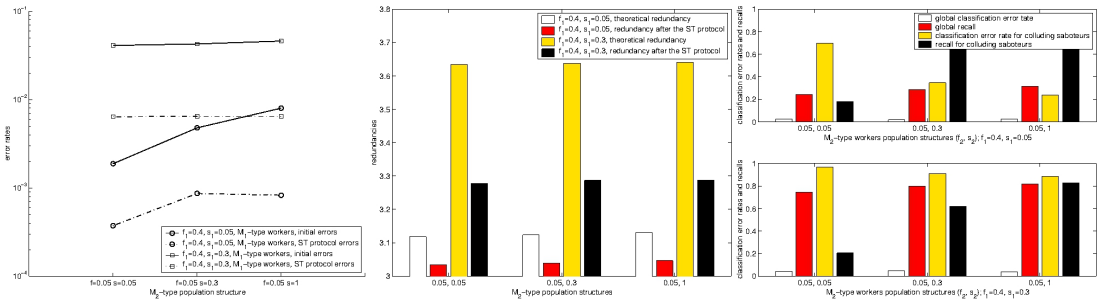


(a) Error rates **(b)** Redundancies **(c)** Classification error rates and recalls

Figure 1.21: Results obtained for a population structure with honest, M_1 and M_2-type nodes, M_1-type workers being in a medium ($f_1 = 0.2$) proportion

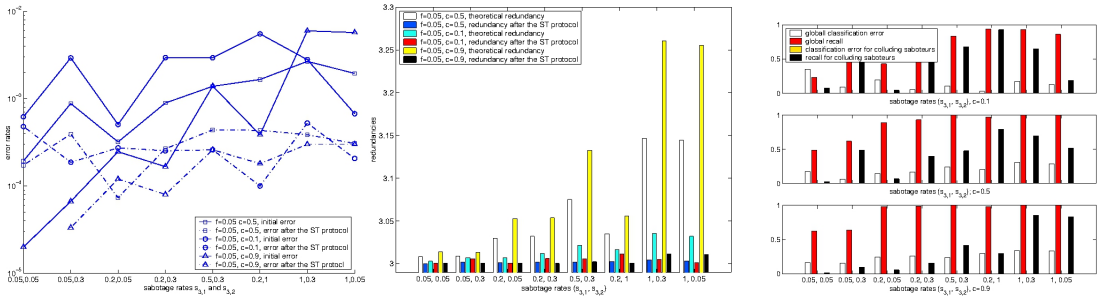
We can notice (figures 1.20a and 1.21a) that if the naive workers do not overwhelm the colluding ones (the cases when $f_1 = 0.05$ and $f_1 = 0.2$), then the ST protocol is very effective in spotting out the collusion, especially on the cases when the colluding workers are well defined (the sabotage rate is big enough). For the case when $f_1 = 0.4$ (figure 1.22a), colluding workers have only a very small influence on the overall and we got a situation similar with a pure M_1 population. Still, we get 10 times improvement in the error rate.

In this mixed case, the redundancy (figures 1.20b, 1.21b and 1.22b) is in between the pure population cases. In majority of cases, the redundancy is small,



(a) Error rates **(b)** Redundancies **(c)** Classification error rates and recalls

Figure 1.22: Results obtained for a population structure with honest, M_1 and M_2-type nodes, M_1 -type workers being in a large ($f_1 = 0.4$) proportion

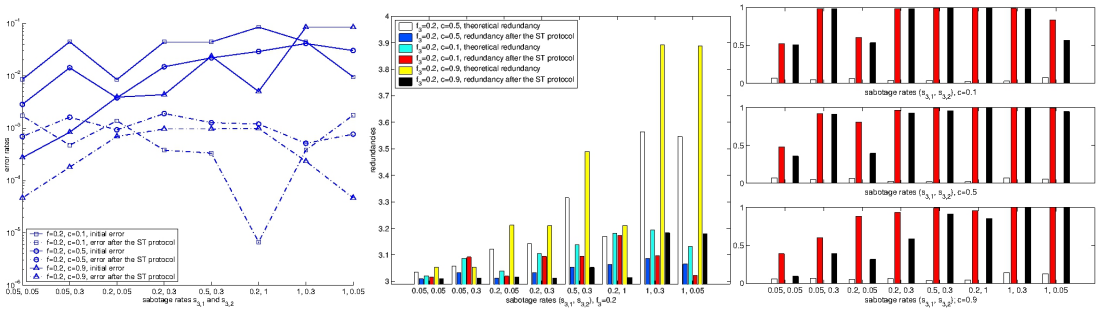


(a) Error rates **(b)** Redundancies **(c)** Classification error rates and recalls

Figure 1.23: Results obtained for a population structure with M_3-type nodes and various naive rates, M_3-type workers being in a small ($f_3 = 0.05$) proportion

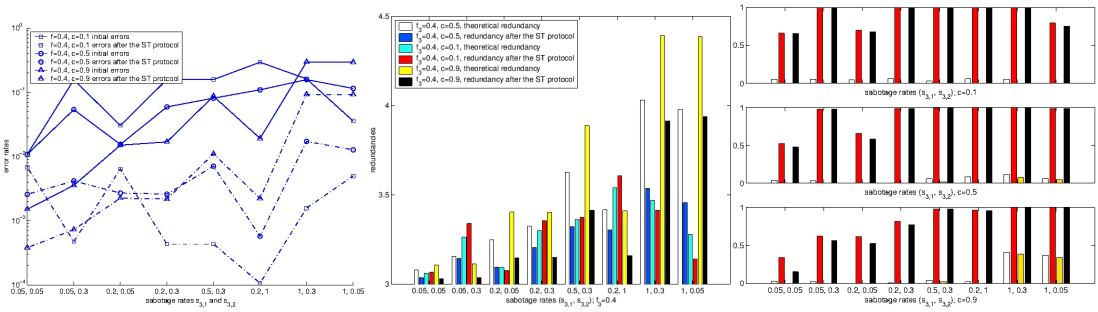
being very significantly below the theoretical one. We can notice that for the case when colluding workers are in a large proportion ($f_2 = 0.4$ - figure 1.20b), if the naive workers shows their profile by a big sabotage rate ($s = 0.5$), the redundancy is lower than the case of only pure M_2-type workers, as the naive saboteurs are spotted out by the procedure described in section 1.4.3.3.

In what regards the classification, the ST protocol correctly identifies most of the colluding saboteurs. We should note that big classification error rates for the colluding saboteurs are reported for the cases when naive saboteurs overwhelm the colluding ones ($f_1 > f_2$). But, this is achieved concurrently with a very low



(a) Error rates **(b)** Redundancies **(c)** Classification error rates and recalls

Figure 1.24: Results obtained for a population structure with M_3-type nodes and various naive rates, M_3-type workers being in an average ($f_3 = 0.2$) proportion



(a) Error rates **(b)** Redundancies **(c)** Classification error rates and recalls

Figure 1.25: Results obtained for a population structure with M_3-type nodes and various naive rates, M_3-type workers being in a large ($f_3 = 0.4$) proportion

global classification error (see figures 1.21c and 1.22c). This means that our ST protocol also identifies naive saboteurs that happened to vote together and classified them as colluding.

As we discussed previously, a M_3-type worker is a hybrid one. We again considered various proportions of M_3-type saboteurs ($f_3 = 0.05$, $f_3 = 0.2$ and $f_3 = 0.4$) and naive rates ($c = 0.1$, $c = 0.5$ and $c = 0.9$) over a full combination of sabotage rates $s_{3,1}$ and $s_{3,2}$ (figures 1.23, 1.24 and 1.25).

The ST protocol results for populations consisting on M_3-type workers ((figures 1.23a, 1.24a and 1.25a)) do not differ too much from the one presented up

to now, being similar with the case of mixed M_1 and M_2 populations. The redundancies show up a similar pattern as explained before. Again, our protocol is much effective when the M_3 saboteurs behave mostly as colluding ones ($c < 0.5$) and have a well defined colluding profile ($s_{3,2} \geq 0.3$). But, also, in the rest of cases, we obtain at least a 10 times improvement comparing with the basic replication. The results of the classification procedures are also very good, the algorithm identifying almost all colluding saboteurs.

Mixing M_3-type workers with pure M_1 and M_2 ones does not change the above results.

1.4.4.2 Discussion

In the Section 1.4.4.1 we have drawn out the following conclusions:

- we succeed to keep the error rate in the acceptable limit of 10^{-4} for the most majority of cases
- if the malicious workers reveal their colluding profile with high consistency (some sabotage rate $s_2 \geq 0.3$), our sabotage tolerance heuristic spots them successfully, even if the number of saboteurs is large
- in all cases, we get at least 10 times improvement comparing with the simple replication, without a meaningful increase of the redundancy. Even in the worst case (with a large number of very effective colluding saboteurs), the redundancy remains below an entire additional replication per work unit.
- the classification procedure is effective, spotting out most of the colluding workers with a low classification error rate.

From the experiments it appears that the most difficult situation for our sabotage tolerance approach occurs when there are many colluding saboteurs (e.g. $f_2 = 0.4$) and when they sabotage very infrequently ($s_2 = 0.05$). Here, our protocol succeeds to lower the error rate, but it still remains around 10^{-3}. If possible, a solution might be to increase the number of voting pools per worker (N). Nevertheless, the actual findings in DGs make us think that such a scenario has a very low probability of occurring in practice. In fact, our selection of N

was balanced between the good results it yields in most cases and the need to keep it within the realistic assumptions of a DG environment.

When presenting the sabotage models in subsection 1.4.2.2 we assumed that the malicious nodes are not aware of our sabotage detection mechanism and they act to conceal themselves from majority voting. Now, let suppose that some 'very' malicious M_2-type nodes are aware of our ST protocol and they will sabotage only when *all* nodes in the voting pools are also M_2-type ones. If *every* M_2-type node in the population is exhibiting such a behavior, these nodes will score no votes against and will be never discovered. This behavior happens to be at the limit of the behavior presented in the paragraph before, with M_2-type nodes sabotaging very infrequently - as the probability the master assigns three M_2-type nodes in a voting pool is very low (0.064 if $f_2 = 0.4$). A solution to tackle such saboteurs is presented in subsection 1.4.5.

We also think that colluding M_2-type nodes aware of our ST protocol and waiting for showing themselves up to the auditing phases of the protocol (steps 8 and 9 from algorithm 2) might not undermine the power of our algorithm. Suppose that M_2-type nodes do not sabotage in the initial round of voting. Therefore, conflicting voting pools will involve only M_1-type nodes which will be spotted out by the classification algorithm. In this case, the number of remaining suspect voting pools will be very low (almost zero) and steps 8 and 9 of the algorithm will almost be avoided. This case is similar with the one that only M_1-type nodes live in the population, besides the honest ones. More, if M_2-types nodes sabotaging in the initial phase are co-existing together with M_2-type nodes that sabotage only in the auditing phases, as the ST protocol selects for auditing honest nodes, it will be very unlikely that two or three such ST protocol-aware M_2-types nodes to be selected for auditing.

Another difficulty of the ST protocol occurs when the number of naive malicious workers is large ($f_1 = 0.4$). The effectiveness of our ST protocol is closely related to the weakness of replication in these situations, as shown by Sarmenta [2002]. In fact, although we succeed to get improvements, to increase performance one might need to increase m.

Another issue with our heuristic concerns the fact that we do not eliminate completely all erroneous results. This results from a number of facts. First, we select "honest" workers to verify the suspicious results. Although we have a very

good confidence that our selected workers are honest, we can not eliminate the possibility of selecting malicious workers instead. This situation can happen with higher probability if the number of saboteurs is very big. Second, the classification algorithm of Section 1.4.3.3 is tuned for a compromise between error classification (false positives) and recall (total number of real positives identified). If we want a very small classification error rate for this algorithm, the recall may be lower and conversely, for a large recall we should accept a larger classification error rate. The larger the classification recalls are, the lower will be the redundancy of the ST protocol and the number of errors in our ST protocol. On the other hand the lower the classification error is, the lower gets the ST protocol error rate. In any cases, 100% recall is not achievable by any possible classification scheme.

Regarding the computational effort, the matrix multiplication algorithm is the most costly part. Kamvar et al. [2003] give a computational analysis for this cost. To compute the initial probability estimates $p_{v,i}$, our algorithm scans once the total voting pool results. To compute the joint distribution functions F_v, the algorithm performs for each worker at most N^2 multiplication and addition operations. To compute the matrix C, the algorithm performs a summation for each pairs of workers (quadratic complexity).

Thus, although the computation is somewhat heavy, the master has to perform only scalar operations with quadratic complexity and this can be run off-line.

Although our algorithm is an off-line one, we do not state that the master has to wait until the whole pool of client requests have been processed. The master simply has to set a proper value for N (like 30 in our experiments) and run the enhanced ST protocol once an average number of N results have been collected per worker. As we mentioned previously, bigger this number is, better performances are obtained. Practically, DG projects with average-to-short task length which usually employ $m = 2$ replication can afford values of N from 20 to 30, which is enough for providing a good protection against collusion. Such values of N are equivalent to running the enhanced ST protocol at every 1-2 weeks, for projects like the ones supported in Einstein@Home or World Community Grid.

1.4.4.3 Classification alternatives

At the heart of the heuristic presented in section 1.4.3 resides the initial classification procedure that identifies with a good reliability the naive M_1-type workers.

We agreed upon the presented statistical approach after we tested several alternatives, offered by the machine learning field, presented in this section. First, we should note that we have an unsupervised learning task with the goal of identifying the naive saboteurs. As presented in section 1.4.3.2, nodes belonging to different types are characterized by different number of votes against collected during their life in the system. Thus, the number of votes against represents the essential information to profile a node.

Literature suggests to apply clustering algorithms like ROCK [Guha et al., 1999], applied to cluster the US Congressmen after their voting behavior during some period of time. If we consider each node represented as a categorical vector of votes per voting pools, our volunteer computing setup differs from the US Congress setup because a node votes only on very few work units from the total number of work units generated in the system. In the US Congress each congressman votes in the majority of open issues. Thus, the distance measure defined by Guha et al. [1999] is not applicable in our case and we got very unstable results with this approach.

For clustering, machine learning suggests k-means [MacQueen, 1967] as a simple heuristic to unsupervised cluster individuals. We tested various representation of our nodes. First, we employed the $P \times P$ matrix of votes against, where P is the number of workers, each node having allocated a row in this matrix. Because this matrix is very sparse, the euclidean distance used by k-means does not have a strong-defined meaning. Thus, k-means succeeds to separate the naive workers only when they sabotage strongly, i.e. $s_1 > 0.5$, which is not acceptable for a desktop grid environment. If each node is represented by its distribution function F_{v_i}, we got somehow better results. But, in both cases, k-means does not guarantee the convergence. For cases with naive workers showing up very infrequently ($s_1 < 0.1$), the classification performance of k-means is very poor. We got the best of k-means for our problem if we represented each worker by the normalized values of the total number of votes against and the total number of voting pools where the node get defeated. In this case, k-means succeeds to extract out a good number of naive workers (high recall), but on the cost of big number of false positives (high error rate). Figure 1.26 shows the classification results, with k-means compared with our statistical approach for several population structures.

Retrieving a high number of false positive naive workers strongly influence the

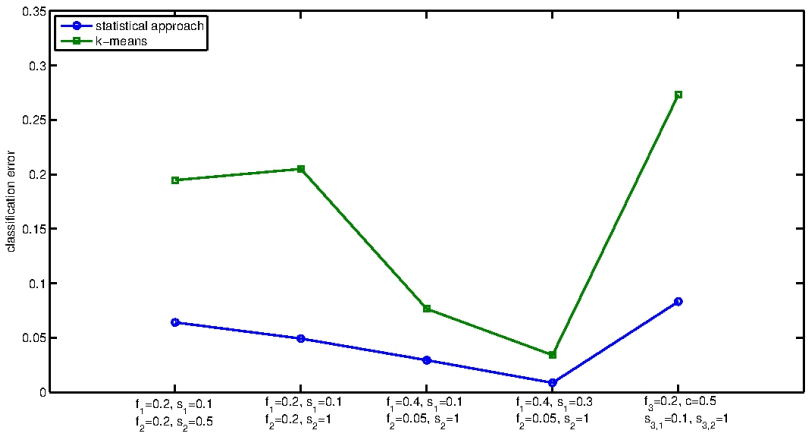


Figure 1.26: Classification error rates for k-means clustering procedure against the statistical approach for various population structures

performance of the auditing procedure of algorithm 2. Thus, there is a higher possibility to select a malicious node to audit a suspect voting pool and less voting pools will be considered suspicious, as the number of nodes classified as honest is lower. Figure 1.27 shows global comparative results, plotting together the error rate of the ST protocol when performing the classification with k-means and with our statistical approach.

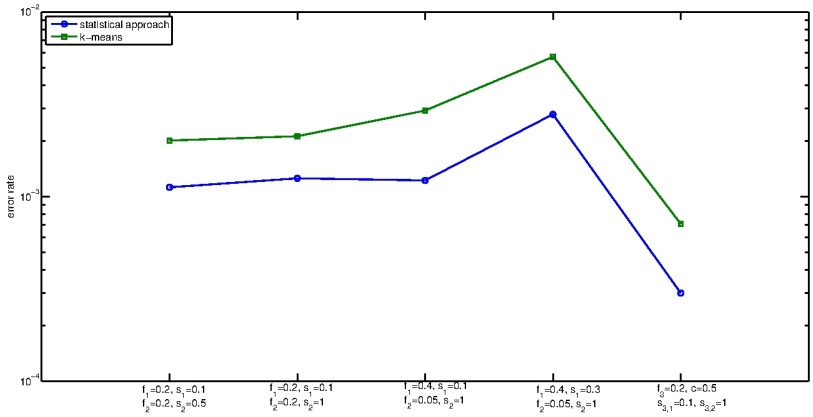


Figure 1.27: Error rates of the ST protocol when using k-means clustering procedure against the statistical approach for various population structures

1.4.5 The Maximum Independent Set Approach

In this section, we shortly brief the Maximum Independent Set approach, developed by Araujo et al. [2011] at University of Coimbra Portugal to overcome the modeling drawback of the M_2 colluding nodes, which were assumed as not being aware that some collusion resistant sabotage tolerance protocol is employed by the master.

This work introduced another type of malicious worker, the M_4-type, which can betray its partnering colluders to cover its course of action. After finding peer colluders and deciding for the sabotage, an M_4 node may retreat and vote correctly. This is useful to overcome a collusion detection mechanisms. Hence, we can say that M_4 nodes take the resistance against collusion detection algorithms a step further than M_2 nodes, because they disguise their behavior even further, by voting against each other.

Sabotaging behaviors defined up to now (from M_1 to M_4) set their decision to sabotage based on the current voting pool alone, i.e., independently of what they did in past voting pools. Unlike this, new malicious M_5-type workers only decide to sabotage when they are sure to be undetectable by an Maximum Independent Set (MIS) algorithm, as defined below. A M_5 node takes the decision to sabotage as following. If three M_5 nodes meet in the same voting pool, they will always sabotage. A single M_5 node never sabotages. The difficult decision occurs when only two M_5 nodes meet in the same voting pool. To tackle this case, each M_5 node keeps a set of peers that voted in coalition against correct nodes. Consider this to be S_1 and S_2, respectively for nodes N_1 and N_2. In the beginning $S_1 = \{N_1\}$ and $S_2 = \{N_2\}$. After N_1 and N_2 vote against a correct node, we define a new peer set $S_1 \cup S_2$, which becomes the peer set for all nodes in $S_1 \cup S_2$ (including N_1 and N_2). If $S_1 \cap S_2 \neq \emptyset$, N_1 and N_2 cannot sabotage together in the same voting pool. The previous scheme ensures that M_5 nodes will always belong to an MIS (given that all the other nodes are correct). Briefly, the idea of the proof is that the votes against graph is a tree (possibly a forest). Whatever root we pick for (each one of) the tree(s), all leafs are M_5 nodes. Also, correct and M_5 nodes do not have edges to their own type of nodes. Therefore, M_5 nodes outnumber correct nodes in this graph, thus forming an MIS.

To proceed with the MIS analysis, workers are arranged in the *votes against*

graph, as in figure 1.28. In this figure, node B voted against A and C, node C against D and B and so on. In a graph, a MIS is the maximal independent set with largest cardinality. Informally, this is the largest set of vertices from a graph such that no edge connects two of those vertices.

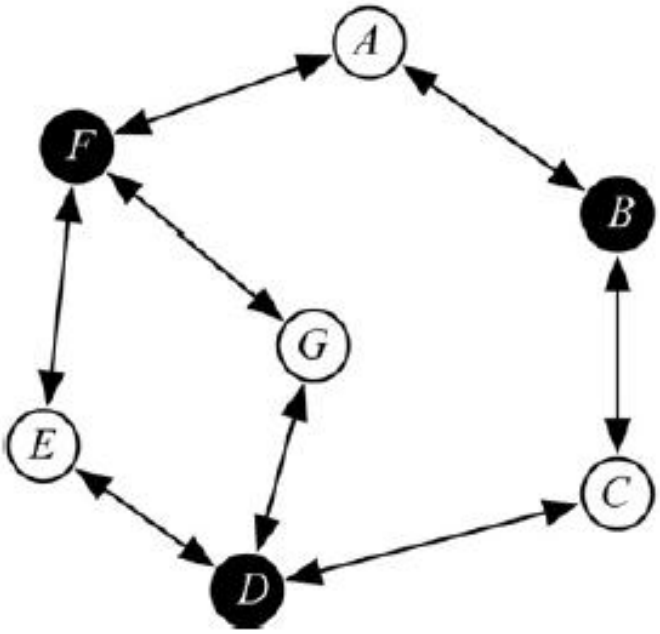


Figure 1.28: The votes against graph

The votes against graph helps to detect the colluders, considering true the statement that the largest plurality of nodes do not vote against each other. In [Araujo et al., 2011], authors proved that MIS can exclude at most b nodes, where b is the real number of malicious nodes. Further, the probability that MIS includes the set B of malicious nodes after having x votes against unknown correct nodes decreases with $\mathcal{O}((\frac{b}{h})^x)$, where h in the number of honest nodes. Therefore, computing MIS can identify the colluders, but this is a NP-complete problem, as acknowledged by the graph theory [Garey and Johnson, 1979].

Araujo et al. [2011] implemented two searching heuristics in an attempt to identify the MIS: the Higher-Order Ratio Heuristic Algorithm [Chang et al., 1988] and the Greedy Randomized Adaptive Search Procedure [Resende et al., 1998]. Compared with KMeans and other variations of searching algorithms based on

votes against, the MIS approaches produce very good results in detecting the colluding nodes group, regardless whether they fit in any of the M_2 to M_5 behavior types. Besides the Sybil attack, already tackled with collusion aware algorithms devised in Silaghi et al. [2009], the MIS approach is able to deter the whitewashing attack, with (anonymous) nodes that leave the community to come back with a new identity. We further computed the gain of the colluders of types M_4 and M_5, showing that they achieve very low gains in the range of $5\% - 20\%$; thus those colluders being actual contributors, as most of their results are valid. In this case, sabotaging colluders would find their effort worthwhile only if they manage to sabotage few crucial results.

1.4.6 Conclusion

In this section we presented our results solving the problem of collusion between saboteurs in desktop grids. The sabotage problem exists when the system is deployed in Internet, with anonymous contributors. Typically, research community [Domingues et al., 2007] approached the sabotage problem by considering that workers are isolated and independent each of another. But, this assumption is not any more valid when P2P concepts are employed for architectural design of the desktop grid system. Within these new conditions, we formally defined several collusion behaviors, fitting here new attacks like the Sybil attack and whitewashers.

To tackle collusion, we designed a statistical toolkit able to classify the honest workers out of the total pool. The classification scheme is based on computing the observed statistical distribution of votes against received by the workers, when participating in the voting pools of the replicated tasks. Identifying the hones workers allow the master to spot out suspect voting pools and apply further auditing on them. Our sabotage tolerance protocol was extensively validated by simulation with all various configurations of the population structure. We show that we effectively tackle the Sybil attack, while still preserving the practical assumptions of a volunteer computing desktop grid. Further, we contributed to the definition of a new sabotage tolerance protocol based on the votes against, which computes the Maximum Independent Set out of the graph of the workers. The MIS approach allows a new clustering of the workers, being able to spot

out the whitewashers. These last sophisticated workers are very effective against reputation-based approaches, because they are able to quit the participation in the system after sabotage and return back under a new identity.

Approaches considered in this section are worthwhile for tackling sabotage in open P2P systems in general. We work on proving that collusion exists in social networks and to apply there our approaches.

1.5 SLA Negotiation in Competitive Computational Grids

Automated and intelligent negotiation solutions for reaching service level agreements (SLA) represent a hot research topic in computational grids. Previous work regarding SLA negotiation in grids focuses on devising bargaining models where service providers and consumers can meet and exchange SLA offers and counteroffers. Recent developments in agent research introduce strategies based on opponent learning for contract negotiation. In this section, we design a generic framework for strategical negotiation of service level values under time constraints and exemplify the usage of our framework by extending the Bayesian learning agent [Hindriks and Tykhonov, 2008] to cope with the limited duration of a negotiation session. We prove that opponent learning strategies are worth for consideration in open competitive computational grids, leading towards an optimal allocation of resources and fair satisfaction of participants.

Contribution of this section was initially presented as [Silaghi et al., 2010] and further elaborated in [Silaghi et al., 2011].

1.5.1 Research objective

The *Service Level Agreement* (SLA) concept represents the key element towards a business ready infrastructure empowering the service economy in a flexible and dependable way [SLA@SOI Consortium, 2008]. A Service Level Agreement is a contractual obligation between a provider and a consumer defining the mutually agreed understandings and expectations - namely the Quality of Service (QoS) values, about the provision of a service [Andrieux et al., 2007; Yan et al., 2007]. In computational grids, SLAs are used to establish service-delivery frameworks between providers and consumers, controlling the usage and the provision of resources [Pichot et al., 2009]. Providing efficient resource allocation in a computational grid is regarded as a complex undertaking due to its scale and the fact that resource owners and consumers may have *different* goals, policies and preferences [Sim, 2010]. When designing the resource management solution, one critical issue is to provide proper mechanisms that induce the involved parties to agree on the QoS values for the established SLAs. Usually this is done in a

negotiation phase which is automated in the infrastructure middleware that takes care of the SLA management.

In computational grids, a fundamental characteristic is the automation of the SLA negotiation process, in the sense that stakeholders rely on the existing middleware to deploy resources in the grid and to consume them. For instance, in grids standardization efforts were undertaken to provide the formal definition of a message protocol - named WS-Agreement Negotiation [Andrieux et al., 2007; Clark et al., 2009; Waeldrich et al., 2011] towards fully specifying the SLA between two parties. In WS-Agreement Negotiation, SLA negotiation is regarded as an exchange of bids between the provider and the consumer up to the final agreement. When applied to open Peer-to-Peer (P2P) service systems, SLA negotiation can become a tool to increase the dependability. Peer-to-Peer systems are vulnerable to various attacks, peers trying to exploit the internal mechanisms for their benefit (e.g. by free riding). All major P2P systems like Gnutella [Adar and Huberman, 2000] or BitTorrent [Cohen, 2003] base their functioning on some economics principles for service exchange, providing the participants with incentive schemes, and trying to avoid the self-interested malicious players undermining the global system stability.

Therefore, if we approach heterogeneous computational grids with open participation, where everyone is welcome to join, provide and consume services, SLA negotiation should be properly devised and carried out to insure several desirable properties like optimal allocation of resources, fair satisfaction of participants, system stability and a high degree of dependability. SLA negotiation might be modeled as a non-cooperative sequential bargaining game [Ausubel and Deneckere, 1993] with self-interested players adopting various strategies. Typically, according with their profile, participants will select one of the available negotiation strategies and contribute to the grid by entering automated negotiation sessions for service delivery and consumption. However, designers should equip the computational grid with robust SLA negotiation strategies that lead to outcomes that fulfill as much as possible the principles enumerated above. The open research question is which are those negotiation strategies that should be deployed in such a system. With proper designed negotiation strategies, the system might deliver outcomes as if a loose collaboration were induced between self-interested competing participants.

In this contribution we tackle the subject of automating SLA negotiation in computational grids, with competing participants and under time constraints. Previous work regarding SLA negotiation in grids [Cheng et al., 2010; Figueroa et al., 2008; Lang, 2005; Lawley et al., 2003; Li and Li, 2004; Li et al., 2007; Li and Yahyapour, 2006; Pichot et al., 2009; Siddiqui et al., 2006; Yan et al., 2007; Zulkernine et al., 2009] focuses on devising SLA negotiation frameworks using bargaining models with agents employing various concession-based strategies. However, negotiation time is not explicitly considered as a resource and opponent modeling is absent. Here, we build a class of intelligent strategies for contract negotiation based on opponent modeling, recently developed in agent research [Hindriks and Tykhonov, 2008], and present a framework for strategical negotiating QoS values under time constraints.

The contribution of this work is two folded. First, our time-constrained framework is characterized by generality, in the sense that it can be adapted to any standard intelligent negotiation strategy that learns the opponent's profile. Time-constrained negotiation is of key importance in a SLA-aware middleware, because, negotiation time is finite and participants can not endlessly wait for obtaining the SLA. Second, we discuss and argue the usage of opponent's profile learning strategies for SLA negotiation. By experimentation, we show that equipping a SLA-based open system with our devised negotiation strategy, we can achieve optimal allocation of resources and fair satisfaction of participants.

Subsection 1.5.2 formally describes the SLA negotiation problem. We start by formalizing the SLA negotiation game. Next, we present the state-of-the-art regarding intelligent negotiation strategies and shortly describe the Bayesian learning agent, further used as a baseline example to develop a time-constrained strategy. Subsection 1.5.3 presents the general framework for building time-constrained negotiation strategies. Subsection 1.5.4 presents the experimental results of a particular time-constrained negotiation strategy based on opponent learning against several other strategies considered by the literature and subsection 1.5.5 concludes the section.

1.5.2 Background and related work

In this section we formalize the SLA negotiation setup and introduce the existing work on negotiation strategies, which constitutes the foundation of our contribution. Moreover, we argue for the need of SLA negotiation strategies in open computational grids, emphasizing that, from economics point of view, an opponent learning negotiation strategy has several beneficial properties for a SLA-based open grid.

1.5.2.1 SLA negotiation formalization

In this subsection we formalize the SLA negotiation setup. We tackle a virtual environment with service providers and users (consumers). Let $X = \{x_1, x_2, \ldots, x_n\}$ denote the set of all services, with x ranging on X. Let SP denote the set of service providers, with sp ranging on SP, and function $S : SP \rightarrow \mathscr{P}(X)$ denoting the services provided by a service provider, where $\mathscr{P}$ represents the power set operator. Let SC denote the set of users (service consumers) of the system, with sc ranging on SC.

Each service has associated issues of interest, denoted by set I, which users are interested in negotiating; variable i ranges on I. Function IS represents the set of issues of interest for a service: $IS : X \rightarrow \mathscr{P}(I)$. Given a SLA-based environment, for a given service, users are interested in negotiating issues like the price, penalty and the properties of the service. Our framework covers both functional and non-functional attributes of a service, given that for each attribute, either it can be defined a finite list of possible options or the service level values are expressed as real numbers.

Negotiation finishes with a contract (the SLA) that is composed of the actual price paid for the service, the penalty and the actual service level values. Function $O^{sc} : X \times SP \times I \rightarrow \mathscr{R}$ denotes the expectation of user sc on the services she uses, where $\mathscr{R}$ denotes the real numbers. Notation $v_{x,i}^{sp,sc}$ represent the expectation of user sc on issue i of service x supplied by provider sp, which in fact, are the QoS values.

In this contribution we focus on the bilateral SLA negotiation, between a service provider sp and an user sc for a given service x. We assume that agents are in place to automate this negotiation, and for each negotiation session we

restrict the available time to T_{max}.

We denote by $U(v) = U_x^{sp,sc}(v)$ the utility that user *sc* gets by obtaining the actual value $v = (v_{x,i_1}^{sp,sc}, v_{x,i_2}^{sp,sc}, \ldots, v_{x,i_n}^{sp,sc})$ of the consumed service. Similarly, $W(v) = W_x^{sp,sc}(v)$ represents the utility that user *sp* gets by delivering the actual value v. From now on, as we deal only with a given service, supplied by a given provider and consumed by a specified user, we will drop the letters *sp*, *sc* and x.

We assume that the utility functions $U(v)$ can be written by a linear combination of the individual utility functions $U_k(v_k)$ (á la Von Neumann [von Neumann and Morgenstern, 1944]):

$$U(v) = \sum_{k=1}^{n} \omega_k^{sc} U_k(v_k), \ \ with \ \sum_{k=1}^{n} \omega_k^{sc} = 1. \tag{1.7}$$

where $U_k(v_k)$ represents the utility that the consumer obtains by receiving the value $v_k = v_{x,i_k}^{sp,sc}$ for the issue i_k and $0 \leq \omega_k^{sc} \leq 1$ represent the weights measuring the importance of a given issue k for the user *sc*. Similarly, utility function $W(v)$ can be written as a linear combination of individual utility functions $W_k(v_k)$ with weights ω_k^{sp}, $0 \leq \omega_k^{sp} \leq 1$.

The negotiation can be formalized using game theoretical foundation. A *non-cooperative bilateral game* is defined by a set of possible strategies for each player (sets S_1, S_2) and the utilities $u_1(s), u_2(s)$ for both players, for each possible combination of strategies $s = (s_1, s_2) \in S_1 \times S_2$. The SLA negotiation game can be seen as a sequential offer/counteroffer *bargaining game*. Such a negotiation is a finite extensive form game in which, at discrete moments in time, one (and only one) of the players is given the opportunity to make an offer which the other player can either accept or reject [Ausubel and Deneckere, 1993]. A strategy of a player in a bargaining game describes what the player should do at each step, for any possible history of the game till that time. In our case, a SLA (which is the vector $v = (v_{i_1}, v_{i_2}, \ldots, v_{i_n})$) can be seen as a possible one round strategy (i.e. bid) for each of the players, while the value that each player gives to it ($U(v)$ and $W(v)$ respectively) is private information. The final agreed SLA is the outcome of the bargaining game. Given the time constraints, the agreed SLA should be obtained before T_{max} elapses, implying that the bargaining game is finite horizon.

When a provider (seller) negotiates with a consumer (buyer), both parties are interested in obtaining those contract values $v = (v_{i_1}, v_{i_2}, \ldots, v_{i_n})$ (i.e. the final

agreed SLA) that maximize their utility functions $U(v)$ and $W(v)$ respectively. This means that at each moment during the negotiation, the player to formulate a bid is maximizing her expected utility function over the remaining bargaining game.

A designer should endow the automated SLA negotiation system with strategies which insure that the bargaining outcome is fulfilling several desirable properties. In what follows, we formalize the desirable properties of optimal allocation of resources and fair satisfaction of participants.

An outcome of a game is *Pareto efficient* if there is no other outcome under which both players are better off, with at least one strictly better off. In our case, a SLA agreement is Pareto efficient if one player cannot strictly increase its utility without diminishing the utility of the opponent. All Pareto efficient outcomes define the Pareto curve (frontier) of the game. Thus, the first goal of a strategy designer in a negotiation game would be to obtain SLA agreements on the Pareto frontier, which is a minimal economic requirement.

A *Kalai-Smorodinsky outcome* represents the Pareto efficient result for which the utilities of both players are proportional with their maximum possible gains. Thus, given that a total utility $u_1 + u_2$ is produced out of a game, the Kalai-Smorodinsky solution enforces an equalitarian (fair) distribution of this utility between the two players. Note that Kalai-Smorodinsky solution is a feature of cooperative environments as well.

We emphasize that, given a game where a provider *sp* negotiates with a consumer *sc* on a set of issues $IS(x)$ of service x, $U(v)$ and $W(v)$ are private information, being established before the negotiation starts. During the negotiation game, both $U(v)$ and $W(v)$ keep constant constant their functional form, while exchanged bids are not influencing the valuation the users give for the service.

From the point of view of a designer of an open computational grid, it is desirable that agents (consumers and providers) are equipped with the same intelligence and, in a negotiation, they should reach a Kalai-Smorodinsky agreement, even if the environment is in fact non-cooperative. If not possible to reach a Kalai-Smorodinsky agreement, bargaining Nash outcome [1] is also acceptable since it

[1] A *Nash solution* to a non-cooperative game represents a strategy profile (s_1, s_2) where, for both players, the following assertion is valid: one player cannot improve on its utility under the Nash solution, given that the opponent does not change its Nash solution strategy. Among all

maximizes the total welfare of the players. Such an automated environment will have the following desirable properties:

- produces a maximum welfare and
- provides a fair distribution of the welfare between the negotiating agents.

Thus, outside agents have economic incentives to participate and stay longer in the open environment, assuring the long-life and stability of the system by exclusively internal and self-regulated means.

1.5.2.2 SLA negotiation in computational grids

In this subsection we present the existing technological background for SLA negotiation in computational grids and how grid computing and agent research deal with strategical negotiation.

Grid computing research focuses mostly on devising software middleware to efficiently support and integrate widely distributed heterogeneous resources. The WS-Agreement specification [Andrieux et al., 2007], using parts of the Web Services Resource Framework (WSRF), describes a Web Services protocol for establishing agreement between two parties such as a service provider and a consumer. It introduces an extensible XML language for specifying the nature of the agreement and agreement templates to facilitate discovery of compatible agreement parties. WS-Agreement Negotiation 1.0 [Waeldrich et al., 2011] extends the WS-Agreement with support for bilateral negotiation of agreement offers and renegotiation of existing agreements, by providing a symmetric message exchange protocol and a simple negotiation state machine. Definition of concrete negotiation strategies is out of the scope for WS-Agreement Negotiation. But, as defined by Clark et al. [2009]; Waeldrich et al. [2011], the protocol clearly establishes the rules of the negotiation game as a bilateral (competing) negotiation with successive rounds of offers and counteroffers, fitting exactly the theoretical

Nash solutions to a non-cooperative game, we can identify the Nash bargaining solution which maximizes the product of the players' utilities $u_1(s_1, s_2) \times u_2(s_1, s_2)$. Such an outcome can be sustained if both players cooperate during the negotiation, which is a feature of the cooperative environments. However, the challenge is to guide self-interested competing participants towards such an outcome.

formalization presented in the previous subsection. Up-to-date, important grid middleware like Globus and UNICORE implemented WSRF, thus, offering the technological ground for SLA negotiation and establishment in a grid service-oriented architecture.

In P2P systems and open grids, several technological models for SLA negotiation exist. Rubach and Sobolewski [2009] propose a SLA-based SERVICEable Metacomputing Environment (SERVME) capable of matching providers based on QoS requirements and performing autonomic provision and deprovision of services. In a P2P grid, organized as a service marketplace where every peer can request a service and providers are heterogeneous, Di Stefano et al. [2009] present a distributed strategy to discover and acquire a demanded QoS. SLA negotiation is part of the QoS distributed discovery protocol and ensures that some SLA agreement is reached out of a chain of offers and counteroffers messages. The specific negotiation behaviors of peers during the SLA negotiation phase is left out of their scope. At a broader level, Brandic et al. [2010] investigate the functioning of a self-manageable cloud, where participants do not have matching SLA templates and a-priori knowledge about negotiation terms and protocols. They describe the VieSLAF framework to facilitate service mediation and negotiation bootstrapping in such clouds. Cheng et al. [2010] present a framework for resource federation in grids, given that nodes are equipped with policies and they possess criteria to accept or not an agreement.

All the above-mentioned research literature proves that technological background for SLA negotiation exists in Grids, P2P and Clouds. While the interaction protocols and message formats are well established and specified, the specific behavior of the participants is left out of the scope in the majority of technical approaches. This paper will focus on analyzing the specific behavior of service consumers and providers and will propose a negotiation strategy with several beneficial properties, adhering to the latest developments in grid research, presented in the paragraphs below.

Recent grid computing research considers strategic negotiation models for automated SLA negotiation. In the simple bilateral negotiation setup, learning strategies were investigated and tested. Li and Yahyapour [2006] consider Q-Learning [Watkins and Dayan, 1992] applied directly for SLA negotiation with time-decreasing utility functions. With Q-Learning, they try to predict the oppo-

nent action, rather than the opponent profile, given that concession-based strategies are adopted by both players. Figueroa et al. [2008] employ the game theoretical modeling to analyze and solve the particular case of the SLA negotiation game having only two rounds. Zulkernine et al. [2009] propose a policy-based SLA negotiation framework where parties can adopt strategies of type conceder or boulware [Faratin et al., 1998], negotiation time being encoded in the utility functions of the players. Lang [2005] investigates the well-being of the grid if nodes are equipped with dynamic negotiation strategies of the same type conceder or boulware. They conclude that neutral strategies are better for the grid as a whole. Our approach depart from those that consider standard agent types like conceder and boulware. We do not encode the negotiation time in the utility functions. We also let our strategy to be as competitive as possible, in order to avoid the players behaviour to be learned. Even if malicious participants would try exploiting the system rules, a regular system participant using our devised strategy will obtain fair SLAs against them.

Besides this simple bilateral approaches, concurrent negotiation setups were investigated, considering possible out-side options available for both parties [Li et al., 2007]. Yan et al. [2007] present a framework for service composition provision, including autonomic SLA negotiation in a setup with multiple alternatives, using concession-based and trade-off schemes, without opponent learning. Siddiqui et al. [2006] consider allocators as brokers mediating and performing negotiation between an initial SLA request and clients with available capacity. Such one-to-many negotiation setups are of larger perspective [Nguyen and Jennings, 2003], being out of the scope of this paper and representing a venue for further research.

For full recent surveys tackling strategical resource negotiation in grids the reader is directed to consult the works of Sim [2010] and Haque et al. [2011].

In general, models for SLA negotiation in grids are neither concerned on the optimality of the solution or the strengths of the proposed negotiation strategy. These models emphasize the automation and the gains that can be obtained in comparison with a non-automated and zero intelligence solution.

Agent research is mostly focused on the formal aspects of the negotiation and proposes more elaborated negotiation strategies. Negotiating agents were developed for the bilateral setup, including the following ones. The zero-intelligent

agent [Gode and Shyam, 1993] investigates the baseline behavior of a random agent. Jonker and Treur [2001] presents an agent devised in-line with the Belief-Desire-Intention (BDI) architecture of the mid-90's agents. Based on this architecture, Jonker et al. [2007] further investigate whether information disclosure during negotiation is valuable for the agents and shows that even with a limited amount of preference information revealed, agents can still obtain significant joint gains in the outcome. Within the same BDI architecture, [Lopes et al., 2004] present a generic framework to test tactics during a negotiation, such as: starting high and conceding slowly, starting reasonable and conceding moderately, respectively starting low and conceding rapidly. We consider that our framework subscribes to the first mentioned negotiation tactic of [Lopes et al., 2004] (starting high and conceding slowly). Raeesy et al. [2007] apply the fuzzy logic to model the negotiation agents and to describe their strategic behavior. A Bayesian learning framework is introduced by Hindriks and Tykhonov [2008] to approximate the user profiles. Negotiation with the help of Q-Learning is investigated by Dearden et al. [1998]; Jian [2008]. Rather than learning the opponent's profile, Faratin et al. [2002] investigate the optimal strategy to perform trade-offs when proposing new bids. Sim [2002] analyzed how various sorts of negotiation strategies perform under time constraints, theoretically proving that agents adopting conservative strategies will achieve higher utilities when negotiation deadline is long, but with higher risk of loosing the deal to other agents. Our devised strategy fits the conservative strategy definition presented in [Sim, 2002] and also incorporates opponent learning, which there it is seen as a possible improvement.

We insist on the difficulty to practically compare these negotiation solutions. Each author of a negotiation strategy devised it within some specific setups and no strategy was ever compared against the others with the same environmental assumptions. To overcome these problems, the Genius negotiator[1] [Hindriks et al., 2009] was developed at the Delft University of Technology, supplying a unified environment for negotiation strategies comparison. Genius allows the definition of negotiation domains with continuous or discrete attributes and of roles (consumer and provider) profiles including the weights ω_k^{sc} (and ω_k^{sp} respectively) for issues,

[1]http://mmi.tudelft.nl/negotiation/index.php/Genius (consulted on 10 august 2011). We thank Prof. Catholijn Jonker from Delft University of Technology Netherlands that provided us a comprehensive introduction to Genius

the individual utilities $U_k(v_k)$ (and $W_k(v_k)$ respectively) for discrete issues and the proportional first-order utility functions for continuous issues. Continuous issues are treated in Genius by discretization - and, further in our paper, we will consider only discrete issues with the additive utility functions presented in eq. 1.7.

However, up to date, only some of the strategies presented above [Gode and Shyam, 1993; Hindriks and Tykhonov, 2008; Jonker et al., 2007] are implemented in Genius. To ease the comparison of our negotiation strategy with the existing literature, we implemented it in Genius (see section 1.5.4).

In the papers originating from agent research, the focus is on learning the opponent's profile and preferences - time restriction is not a major concern, and, given this, to devise negotiation strategies for mutual benefit of both parties. In our paper we focus on devising a time-based strategy, that in conjunction with an opponent learning procedure, brings further advantages and gains from the negotiation process. Furthermore, the negotiation under time constraints makes the strategy suitable for the practical requirements of a SLA-based computational grid.

1.5.2.3 The Bayesian learning agent

An opponent learning agent, during the play of the game, tries to infer the utility function of the other player. In general, at each negotiation step, such an agent performs two actions:

- analyze the incoming opponent's proposal and update the opponent's profile;
- select and propose the next bid.

Learning the opponent's profile consists mainly of approximating the opponent's utility function [1] $U(v)$. While we assume that the utility function $U(v)$ is a linear combination (see eq. 1.7), the opponent profile is composed of the individual utility functions $U_k(v_k)$ and the weights ω_k, for all issues $k = \overline{1,n}$. Bayesian

[1] During this subsection we adopt the notation $U(v)$ referring to the utility function of the opponent, regardless her role of being a buyer or a seller. The notation $U_{own}(v)$ is used in several places to denote the utility of the agent that is to move and decides what to do next.

learning [Hindriks and Tykhonov, 2008] associates a probability for each possible hypothesis about the opponent profile; a hypothesis being a combination between a vector of weights and the individual utility functions.

As the search space for the opponent profile is infinite, different learning models propose various techniques to reduce it and to make it computer tractable via simplification. Bayesian learning does not precisely learn the weights ω_k of the opponent. While the literature [Faratin et al., 2002] argues that learning the ranks of the weights is typically sufficient to significantly increase the efficiency of the outcome, the search space for the weights reduces to the $n!$ possible rankings of the preferences for the n issues.

For each individual utility function $U_k(v_k)$, the agent preferences for the issue values can be encoded starting from three basic function shapes:

- downhill shape: minimal issue values are preferred over other issue values and the evaluation of issue values decreases linearly when the value of the issue increases;
- uphill shape: maximal issue values are preferred over other issue values and the evaluation of issue values increases linearly when the value of the issue increases;
- triangular shape: a specific issue value somewhere in the issue range is valued most and and evaluations associated with issues to the left and right of this issue value linearly decrease.

A function with a unique extreme point on an interval can be approximated with a linear combination of *basis functions*[1] from the three types presented above. Therefore, given that the domain of an issue k has m_k values, one can draw m_k functions with the maximal issue value in each of the m_k values and linearly combine these functions to obtain the individual utility function U_k. This approach is very suitable for discrete issue domains, while continuous domains might be discretized (i.e. split in m equal subintervals). Thus, for an issue k, the

[1]In mathematics, a basis function is an element of a particular basis for a function space. Every function in the function space can be represented as a linear combination of basis functions, just as every vector in a vector space can be represented as a linear combination of basis vectors (G. Strang, "Wavelets", American Scientist, Vol. 82, 1994, pp. 250-255.)

search for the individual utility function U_k is in fact reduced to learning some numeric values $0 \leq p_i \leq 1,\ i = \overline{1, m_k}$, which allows to compose U_k as a linear combination of triangular functions.

From the discussion above, we note that learning the opponent profile means selecting the proper ranking of weights and assigning some probabilistic values for each triangular valuation function for each issue value, for all issues.

Therefore, in the Bayesian learning setup, a user (opponent) profile is a matrix that associates probabilities for each *hypothesis* $h_j \in H$, where $H = H_\omega \times H_1 \times H_2 \times \ldots H_n$ is the hypotheses space. We note that the hypotheses space lists all possible combinations between the possible preference orders for the issues of the opponent and possible individual utility functions for each issue. One row of this matrix represents the probabilities $(p_\omega, p_1, \ldots, p_n)$ associated for a hypothesis h, and the matrix has $M = n! \times m_1 \times m_2 \times \cdots \times m_n$ lines (the total number of possible hypotheses).

Bayesian learning starts with some random initialization of the agent profile matrix. At each step, this matrix is updated using the Bayes rule. If b_t is the newest received bid, then

$$P(h_j \mid b_t) = \frac{P(h_j)P(b_t \mid h_j)}{\sum_{k=1}^{M} P(h_k)P(b_t \mid h_k)} \tag{1.8}$$

Hindriks and Tykhonov [2008] shows how to compute eq. 1.8, given the setup described above. It is noteworthy that if the opponent agent is a rational one, after a big number of proposed bids, the agent profile matrix converges.

The second action that a negotiating agent performs in each step is *to select and propose the next bid*. In general, a strategy defines some *conditions* that prospective bids should fulfill in order to be included in the subset with acceptable bids to be proposed.

For each possible own bid $b = (v_1, v_2, \ldots, v_n)$ the agent can compute the estimated utility of this bid for the opponent:

$$U(b) = \sum_{j=1}^{M} P(h_j) \sum_{k=1}^{n} \omega_k U_k(v_k) \tag{1.9}$$

A smart strategy is to select and propose that bid that maximizes the utility U_b of the opponent, while the own utility U_{own} to stay as close as possible to the

own target negotiation objective τ:

$$b = \underset{b' \in \{x || U_{own}(x) - \tau| \leq \delta\}}{\arg\max} U(b') \tag{1.10}$$

Agent research [Faratin et al., 2002; Hindriks and Tykhonov, 2008] considers this strategy valuable because, at each step of the negotiation, it increases the chance that the own bid to be accepted by the opponent, without deviating much from the target utility. In our framework defined in subsection 1.5.3.1, we improve exactly this condition, in order to create different concession levels.

Because the Bayesian learning agent should react in a *reasonable time* and both actions to be performed at each step include high running complexity, code and computation optimizations are required to make the agent responsive. Hindriks and Tykhonov [2008] propose some optimizations and show the power of the Bayesian agent against other negotiating strategies, including trade-off [Faratin et al., 2002] and ABMP [Jonker et al., 2007].

We note several drawbacks of the Bayesian agent:

- it does not learn the negotiation strategy of the opponent, as, at every step, it updates the opponent profile only with the information about its *last* bid, never considering the succession of the opponent's bids;
- the agent is highly dependent on the initialization step. At the beginning of the negotiation, the opponent profile is very weak and the agent will propose highly suboptimal bids. Thus, if the negotiation finishes in the first rounds, the Bayesian agent is highly likely to lose;
- as devised by Hindriks and Tykhonov [2008], the Bayesian agent is a concession agent, it linearly decreases the predicted utility value for the opponent over the time. Thus, the agent has more chances to finalize the negotiation in few rounds, which raises the drawback presented just above.

To overcome these drawbacks and to improve the agent as much as possible, suited for a negotiation under time constraints, we propose the framework presented in section 1.5.3. One should note that this framework is general enough to be used in relation with other opponent learning strategies.

1.5.2.4 Why opponent learning agents for SLA negotiation? - an economic point of view

In this subsection we argue, from an economical point of view, for the need of opponent learning agents to negotiate SLAs in open environments. By means of experiments, we emphasize the feature that intelligent agents which learn the opponent profiles bring in the beneficial properties we desired for the whole computational grid environment (i.e. optimal allocation of resources and fairness).

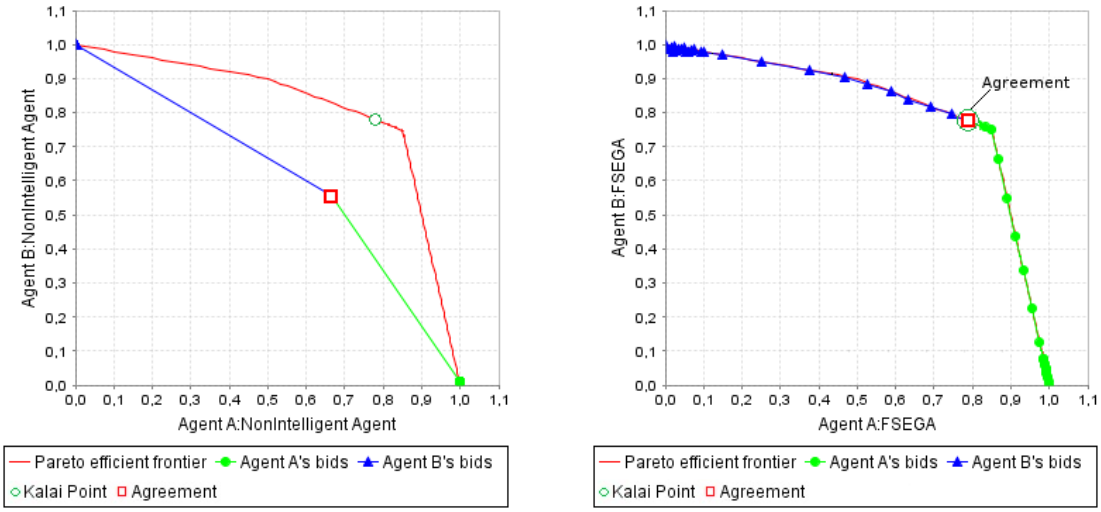
Let's experiment[1] a negotiation between two basic non-intelligent agents on the SON domain[2] [Faratin et al., 2002]. Each agent, either consumer or provider, only records the list of the opponent's bids, without making any inference about the opponent's profile. When proposing the next bid, this non-intelligent agent uses the same condition like in eq. 1.10, but with a narrow target search space, restricted to the list of the opponent's previous bids. In figure 1.29a, we notice that such two agents do reach a sub-optimal agreement, the outcome not being on the Pareto frontier.

In figure 1.29b we let two Bayesian agents (as described in the previous section) negotiate on the same domain. We notice that the agreement moves up to on the Pareto frontier and very close to the Kalai-Smorodinsky solution. Thus, the agreement becomes Pareto-optimal and the solution is very close to the equalitarian cooperative one. In the conclusion, equipping the agents with intelligence will make the self-interested opponents to "cooperate", which is a desired property we induced in the system.

For the design purposes of an open computational grid, the property presented in the previous paragraph is of main importance. If the software designer equips the system with the same intelligent strategic behavior - like the Bayesian learning strategy, for both the providers and the consumers, the usage of such strategies will induce an optimal and fair resource allocation inside the system, given that the QoS properties of the services and the utility functions of the participating peers are a-priori established. We acknowledge that other strategies might exist in order to achieve the optimal allocation and fairness, but, as acknowledged by Sim [2002], the usage of opponent learning is a research alternative towards

[1] All negotiation sessions presented in this paper are implemented and run in the Genius negotiator, previously introduced in section 1.5.2.2

[2] This domain will be further introduced in section 1.5.4



(a) Non-intelligent agents

(b) Intelligent agents

Figure 1.29: Comparison between: (1.29a) a negotiation game between non-intelligent agents, and (1.29b) a negotiation game with intelligent agents that learn the opponent's profiles

this. As we will see in subsection 1.5.4.2, even when the Bayesian learner plays against other strategies on various sorts of negotiation domains, the results are not penalizing the Bayesian.

1.5.3 The time-constrained negotiation strategy

In what follows we present our strategy for negotiation under time constraints. First, in subsection 1.5.3.1 we present the general setup for constructing learning agents, given that a time constraint is imposed for a negotiation session and we show how we instantiated a Bayesian learning agent to accommodate this general setup. We end this section by discussing about the performance estimation of our negotiation strategy on various domains and presenting some mandatory code optimizations.

1.5.3.1 The General Framework

We assume an automated bilateral negotiation, with exchange of bids between the user agent (the consumer) and the service provider agent. A bid b is in fact a SLA

proposal, issued at time t. Negotiation should finish before time T_{max}. If the user and the provider are not able to establish an agreement before T_{max}, they gain nothing from the negotiation game. We restrict the scope of our framework for a single negotiation game, and we do not consider several successive negotiation sessions of the same player.

The agent designed in this section is a learning agent of the kind described in subsection 1.5.2.3, performing two actions at each step: (i) updates the opponent's profile and (ii) select and propose the next bid. Our agent can play both roles, either user (buyer / consumer), or service provider (seller). Although we design our agent for the Bayesian learning setup, we do not restrict the agent design to this specific learning scheme, nor do we restrict the agent for a particular negotiation domain.

In game theory [Ausubel and Deneckere, 1993], in an offer/counteroffer bargaining game, if the game ends at some moment t, the utilities of the players are discounted by some factor such as e^{-rt}. In our case, opposite from previous work [Lang, 2005; Li and Yahyapour, 2006; Yan et al., 2007; Zulkernine et al., 2009] in SLA negotiation in grid computing, there is no clearly specified discount factor. However, the concession behavior may also be understood as the player being conscious that, as time passes, the probability to reach the end of the game without agreement and with zero utility increases.

The main idea of our negotiation strategy consists in adapting the agent behavior to the duration of the negotiation. Usually, a learning scheme takes time to converge to some consistent opponent's profile, therefore the agent should consider to allow several bilateral bids exchange rounds without much own utility concession and concede only when the total negotiation time T_{max} is about to elapse.

Our strategy is to divide the negotiation time $[0, T_{max}]$ in k subintervals $I_1, I_2, \ldots, I_k$, $I_1 = [0,t_1]$, $\ldots$, $I_k = (t_{k-1}, T_{max}]$ and let the agent play different strategies on each subinterval. During all negotiation time, the agent applies the baseline learning scheme in order to estimate the opponent's profile. The strategies played on each subinterval $I_1, I_2, \ldots, I_k$ differ on the step when the agent selects and proposes the next bid such as follows:

- Conditions C_1 for selecting the next bid in subinterval I_1 are the strongest, the agent making very few concessions at the beginning of the negotiation;

- Conditions C_{t-1} for selecting the next bid in the subinterval I_{t-1} are stronger than the conditions C_t for selecting the next bid in the subinterval I_t, for every $t = \overline{2,k}$;
- Conditions C_k for selecting the next bid in the last subinterval I_k are the baseline conditions of the learning scheme.

It results that $C_1 \supset C_2 \supset \cdots \supset C_k$.

In our case, we selected $k = 3$ and divided the interval $[0, T_{max}]$ into three subintervals:

- interval I_1: from $[0,t_1]$, the agent makes only very small concessions;
- interval I_2: from $(t_1,t_2]$, the agent performs like in I_1, but we relax some conditions for selecting the next bid;
- interval I_3: from $(t_2, T_{max}]$, the agent concedes like in the standard setup imposed by the baseline learning agent. Our agent will concede down to a reservation utility, although the agent should be better off with any gains from the negotiation if the total negotiation time elapses without any agreement.

When designing our agent we selected the interval limits t_1 and t_2 by experimentation and devised the conditions for when and how to make concessions.

We introduced the second interval I_2, because we noticed that (see figure 1.30a), without this interval, the agent switches very abrupt from a powerful agent that does not concede to an agent that concedes very quickly. If a smart agent is considered as the opponent, this agent can simply learn this behavior and our agent will never win in this case. By performing the transition between the strong-holding agent behavior of I_1 to the standard behavior of I_3 via some "mixed" behavior in I_2, we harden the job of an opponent to learn our strategy.

The reader can also note that on I_3, at the end of the negotiation, we do not accept an agreement below our reservation utility u_r. This can be: a) the utility of the bid that maximizes the estimated opponent's utility on the whole domain; or b) the utility given by the first offer of the opponent; or c) another of its past offers. Any of these choices should give us an acceptable positive utility, ensuring

as well that the opponent will accept it, thus avoiding the negotiation to end without an agreement.

We note that the time-constrained negotiation framework obtained by the division of the negotiation interval in k subsets is general enough to be particularized to a specific learning procedure, considering the more or less available conditions for the next bid selection. In general, we recommend setting the subinterval limits $t_1, \ldots, t_{k-1}$ as close as possible towards T_{max}, making the agent as inflexible as possible till very close to the ending time. Thus, the agent will develop a conceder behavior only at the end of the negotiation time, being very strong-holding up to then.

The particular application presented in here is an opponent model based agent, which means that it estimates the opponent's utility and use the estimation to propose a bid with reasonable utility for the latter. The learning method used to estimate the opponent's preferences is Bayesian. The preferences' estimation allows the evaluation of the opponent's utility for any possible bid. Remember from Section 1.5.2.3 the general form of a decision of our agent, at a given point in time; that is, to propose a bid that maximizes the utility U_b of the opponent, while its own utility U_{own} varies in an interval $(u_{min}, u_{max}]$:

$$b = \underset{b' \in \{x | U_{own}(x) \in (u_{min}, u_{max}]\}}{\arg\max} U(b') \tag{1.11}$$

The method by which $(u_{min}, u_{max}]$ is updated on a given time interval, generates less, or more concessions from our agent, for the time interval in discussion. The baseline for the update of $(u_{min}, u_{max}]$ is the following: $u_{min} = u_{max} - \delta$, and at each offer, if our bid is refused, then the upper bound of the interval (and implicitly the lower bound) is decreased. The starting value u_{max} of the upper bound is represented by the highest utility of the agent, and then it is sequentially decreased. Notice that u_{max} takes the place of the targeted utility τ from eq. 1.10, from which we accept a deviation of a maximum δ.

Next we explain in detail the strategies of our agent (i.e. the decision rules to offer a bid plus the method of updating $(u_{min}, u_{max}]$) for each of the three particular time intervals considered by our application:

Interval I_1 (btw. 0-85% of the total time): Supposing that b_t is the last

bid proposed by the opponent, then at step $t+1$ the own agent will propose a bid b_{t+1} that not only maximizes the opponent's utility, but also ensures that our agent does not attain an utility lower than the one implied by the opponent's offered bid b_t. Mathematically:

$$u_{min} = \max\{u_{max} - \delta, U_{own}(b_t)\} \tag{1.12}$$

Notice that on this time interval, the negotiation game may end only with the opponent accepting an offered bid; our agent would never accept a bid offered by the opponent.

Interval I_2 (btw. 85-95% of the total time): The conditions for selecting our next bid are relaxed to the usual $u_{min} = u_{max} - \delta$. Our agent accepts the offer b_t only if $U_{own}(b_t) > U_{own}(b_{t+1})$.

Interval I_3 (btw. 95-100% of the total time): The concessions our agent is ready to make increase dramatically. Denoting with u_r the reservation utility for our agent, the update of the lower limit u_{min} will be given mathematically by:

$$u_{min} = \min\{\max(u_r, u_{max} - \delta), e^{c+d(T_{max}-t)}\} \tag{1.13}$$

where T_{max} is the time constraint of the negotiation, $c = \ln(u_r)$, $d = \frac{\ln(\frac{u_{0min}}{u_r})}{(1-0.95)\cdot T_{max}}$, and u_{0min} is the last lower bound before entering in the time interval I_3.

Notice that for $t = T_{max}$, then

$$e^{c+d(T_{max}-t)} = u_r \tag{1.14}$$

which is the minimum of the given exponential function. For $t = 0.95 \cdot T_{max}$, then

$$e^{c+d(T_{max}-t)} = u_{0min} \tag{1.15}$$

which is the maximum of the given exponential function. The rest of the conditions for selecting our next bid, or accepting an offered bid, remain the same as before.

It is straightforward to prove that the set of conditions for selecting the next bid are less and less restrictive as we move from interval I_1 to the interval I_3 (provided that δ is not too big). Therefore, the requirements of the general setup are fulfilled.

1.5.3.2 Performance optimization

In this subsection we discuss several optimization issues to make the negotiation strategy presented above usable in practice.

First, we need to note that, the Bayesian learning algorithm represents the core of the agent, which implies computing a huge matrix with probabilities. The size of this matrix was presented in subsection 1.5.2.3 and it contains $M = n! \times m_1 \times m_2 \times \cdots \times m_n$ lines (the total number of possible hypotheses). E.g. for a domain with $n = 7$ issues and each issue having 7 nominal choices, results that M is greater than 4 billions. The basic Bayesian learning updates this matrix at each step, thus, it results that the agent will spend a lot of time to compute the next offer. During learning, we can notice that the probabilities $P(h_j \mid b_t)$ computed with the help of eq. 1.8 are very small (close to zero) for those hypotheses h_j which are far away from the opponent's real profile. Therefore, a first optimization is to compute the Bayesian probabilities $P(h_j \mid b_t)$ only for those hypotheses h_j which together account for more than 95% of the sum $\sum_{k=1}^{M} P(h_k)P(b_t \mid h_k)$. This needs an additional ordering of the hypotheses, but it highly reduces the number of rows in the Bayesian matrix. E.g. on a domain with $n = 7$ issues, each issue having 7 nominal choices, the Bayesian learner converges after about 7% of the total negotiation time and the relevant hypotheses are only 5% of the total hypotheses space. Thus, this optimization speeds up the reasoning with about 20 times.

With this optimization, we succeed to speed up our agent, which reasons much at the beginning and starts computing faster the Bayesian matrix after this matrix begins to converge. Therefore, within an external imposed time limit T_{max} we succeed to accommodate a higher number of own bids.

Second, given that our Bayesian learner performs quicker as time passes, after every bid we can compute a value t_o which is the average of the time elapsed up to the moment per number of bids. t_o can be used to estimate the number of remaining bids up to the end of the negotiation, in the condition that the pace of the opponent bids remains the same. In this case, if the estimated number of bids on the third interval I_3 is smaller than a threshold, we can dynamically update the interval limits t_1 and t_2 letting our agent to fit at least several bids in the last interval. This optimization is required especially on huge domains where the initial time of thinking per bid is very high and might happen that very few

moves to enter in the last interval. This situation should be avoided, because it can lead that the designed strategical behavior at the end of negotiation not to take place and the session to finish without an agreement.

These two optimizations should be considered together with the code optimization presented in [Hindriks and Tykhonov, 2008], which shows how to optimize the Bayesian agent to the huge memory requirements, given that the agent stores explicitly the list of the hypotheses.

1.5.4 Experimental Results

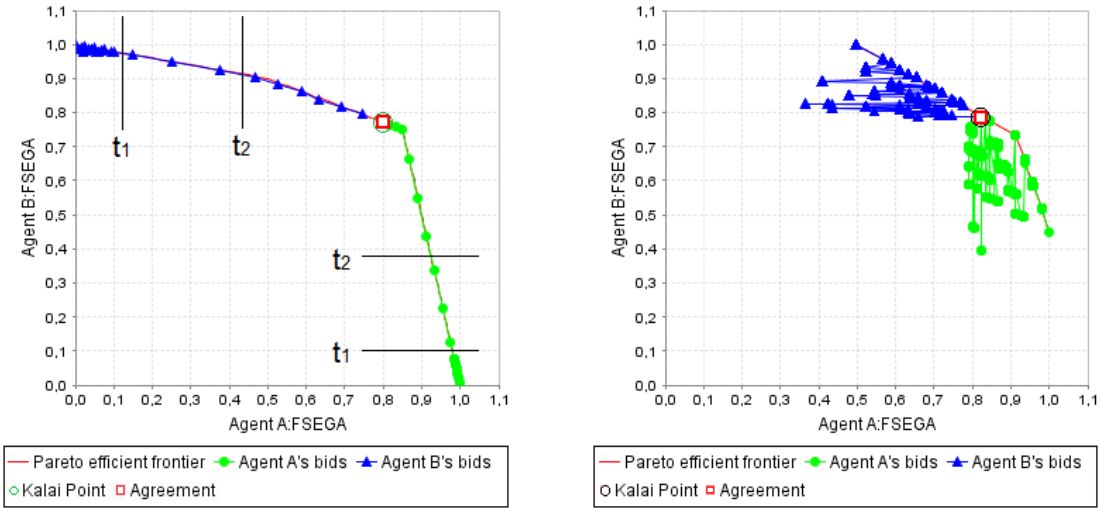
We implemented the time-constrained negotiation strategy with 3 subintervals presented in subsection 1.5.3.1, using the Bayesian learning scheme as baseline in the Genius negotiator [Hindriks et al., 2009]. We named our negotiation strategy AgentFSEGA.

We have run two batches of experiments. First, we tested our negotiation strategy against the ABMP [Jonker et al., 2007] and the standard Bayesian [Hindriks and Tykhonov, 2008] agents, both publicly available in the Genius negotiator. With this experiment, we show the advance of the reasoning model when the negotiation time is considered as a constraint.

Second, we investigated the range of the possible negotiation domains and experimented our agent against the non time-constrained agents used in the first set of experiments and against other negotiation agents available in Genius which were especially designed for a game with time constraints.

1.5.4.1 AgentFSEGA against non time-constrained negotiation strategies

As a testing domain we selected the service oriented negotiation [Faratin et al., 2002] (SON), which fully complies with the SLA formalization presented in section 1.5.2.1. The service described in the SON domain has four issues of interest, each having 30 possible values. Thus, there are 810000 total number of possible agreements. The SON domain is huge, in the sense there are a lot of possible agreements and it requires a very scalable agent. We also tested our agent on another engineered domain which has only 576 possible agreements, but with a stronger opposition between payoffs received by the negotiating parties (there



(a) The SON domain

(b) The small engineered domain

Figure 1.30: AgentFSEGA against itself

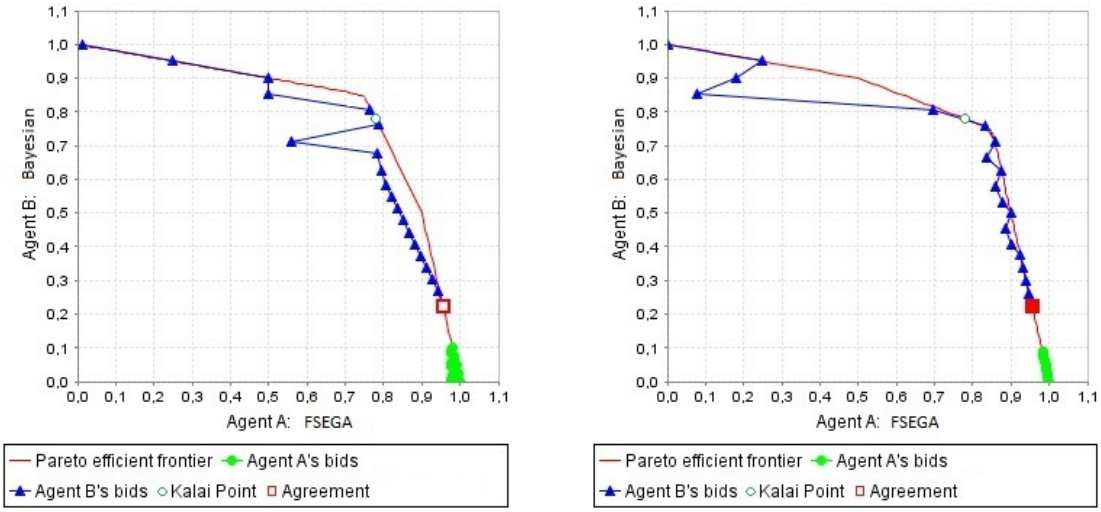
are fewer bids mutually acceptable for both parties, thus it is harder to get an agreement). We set the time limit for a negotiation game to 3 minutes.

In figure 1.30 we depict AgentFSEGA negotiating against itself on both domains. On each axis, we depict the utilities for the negotiating agents. On both figures we depicted the Pareto frontier (with simple line), service provider's bids (with strong circled line) and the opponent's bids (line with triangles). The empty square represents the agreement and the empty circle represents the Kalai-Smorodinsky point. In figure 1.30a we also depicted how AgentFSEGA produces bids as the time elapses.

We observe that (figure 1.30a), at the beginning of the negotiation, AgentFSEGA has a strong-holding behavior, supplying bids in the closer vicinity of its best SLA. More than half of the concession is made only at the end on the negotiation, in interval I_3 after t_2. We see that interval I_2 assures a smoother transition to the concession behavior, as we envisaged in subsection 1.5.3.1.

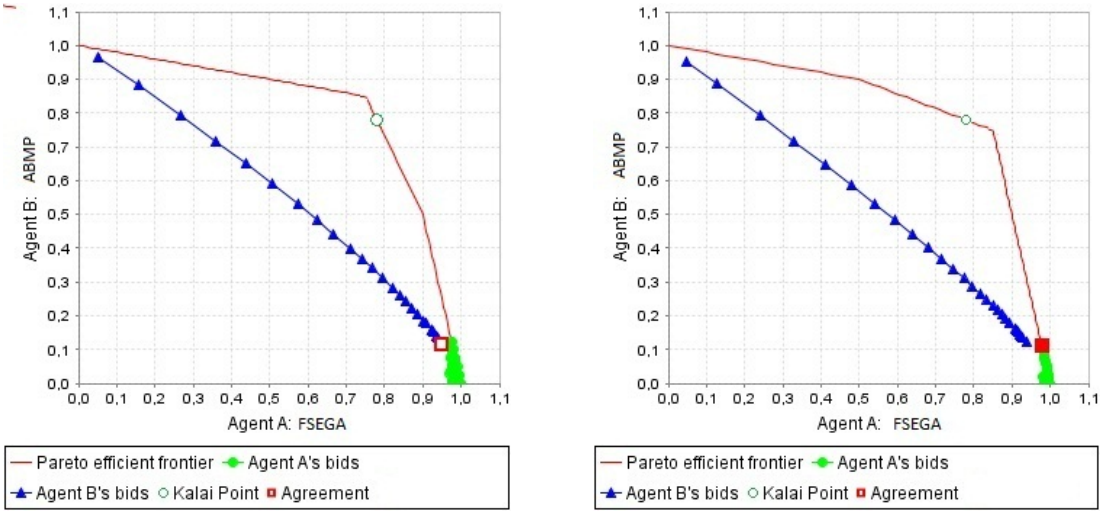
We note that the agreement is established on the Kalai-Smorodinsky point, in which welfare is maximized and the distribution of welfare is fair. Thus full cooperation is induced by the game, although the players are self-interested, with competing utility functions and they do not explicitly cooperate.

Concluding, if both the user and the provider negotiate with similar intel-



(a) AgentFSEGA as service provider **(b)** AgentFSEGA as user

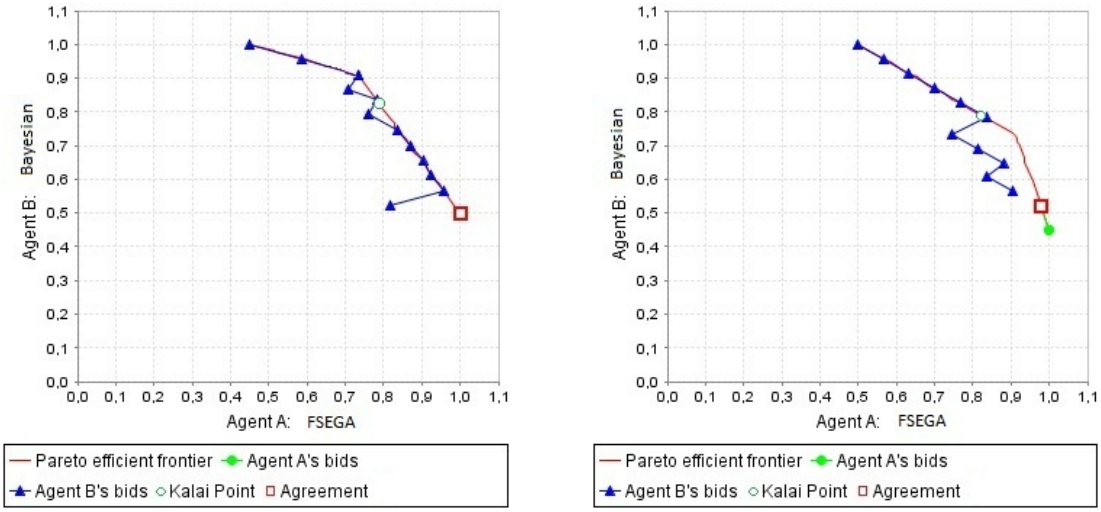
Figure 1.31: AgentFSEGA against Bayesian on the SON domain



(a) AgentFSEGA as service provider **(b)** AgentFSEGA as user

Figure 1.32: AgentFSEGA against ABMP on the SON domain

ligence, i.e. the same representation of the environment and using the same opponent modeling technique, the welfare is maximized - therefore it is valuable to equip a SLA-based environment with such sort of intelligence. This finding also validates the correctness of our agent implementation.



(a) AgentFSEGA as service provider **(b)** AgentFSEGA as user

Figure 1.33: AgentFSEGA against Bayesian on the small engineered domain

Figures 1.31 and 1.32 present the result of AgentFSEGA playing as a user or provider against the Bayesian and ABMP strategies on the SON domain. This experiment models the case of a SLA-based environment with competing stakeholders of different intelligence. We note that in both cases, AgentFSEGA does not concede at the beginning of the negotiation.

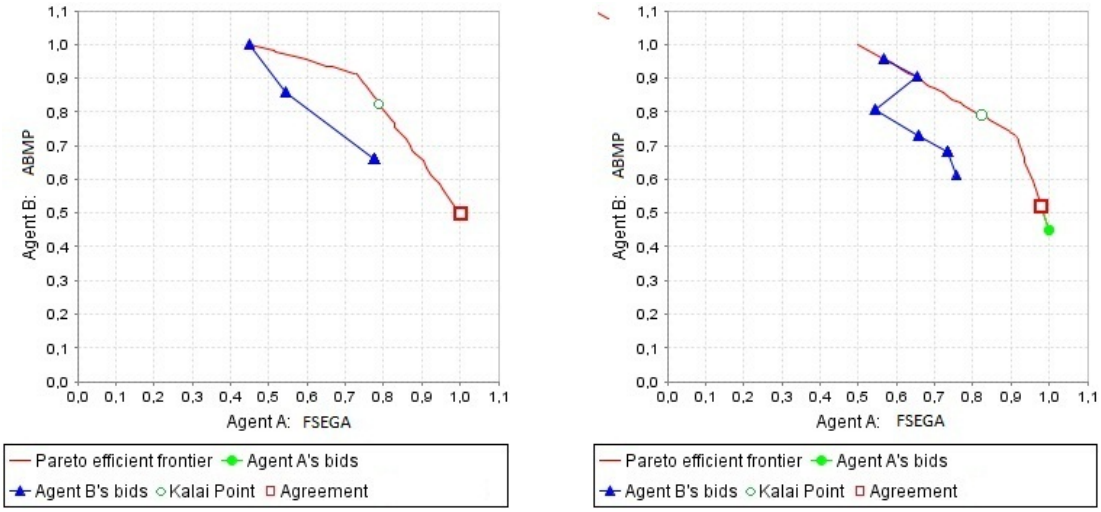
Similar findings result from the small engineered domain (figures 1.33 and 1.34), even if the agreement point is harder to be obtained, i.e. higher number of rounds are needed to reach the agreement.

Both experiments show that the time-constrained agent surpass the baseline agent and other non-time constrained strategies. This emphasize our judgment that letting the learning strategy to converge is worth to consider, given that some finite T_{max} negotiation time is available.

1.5.4.2 AgentFSEGA's performance analysis

In this section we discuss the performance of our negotiation strategy, considering a wide range of domains and agents devised for a competition[1][Baarslag et al.,

[1]We acknowledge the tremendous help from the the Automated Negotiating Agents Competition (ANAC) 2010. ANAC 2010 competition took place during AAMAS



(a) AgentFSEGA as service provider **(b)** AgentFSEGA as user

Figure 1.34: AgentFSEGA against ABMP on the small engineered domain

2012] with time limits.

The size of a negotiation domain is given by the number of issues and the number of discrete values for each issue: $SZ = m_1 \times m_2 \times \cdots \times m_n$. SZ represents the search space containing all possible agreements. For the case of our negotiation strategy, $M = n! \times m_1 \times m_2 \times \cdots \times m_n$ represents the total number of hypotheses space of the opponent. Thus, the bigger SZ, the harder is the negotiation process in general, and the more difficult it is for AgentFSEGA to learn the opponent's profile. In table 1.3 we present the characteristics of several domains considered in this section. Figure 1.35 represents the SLA search cloud for the domains of table 1.3. The small engineered domain, Itex-Cypress and Travel[1] were used during ANAC 2010. We can observe that the first two domains are small ones, while Travel has 7 issues and SZ is 188160.

Clouds of figure 1.35 are depicted considering a given specification for the

2010 conference. Further details can be obtained from the competition web page: http://mmi.tudelft.nl/negotiation/index.php/ANAC_2010 (consulted on 9 August 2011). ANAC 2010 introduced several new agents in the Genius repository, especially designed considering the negotiation time as a constraint.

[1]Further details about the domains can be found consulting the ANAC 2010 paper [Baarslag et al., 2012].

Domain name	Number of issues	*SZ*	*M*
Itex-Cypress	4	180	4320
The small engineered domain	5	576	69120
SON	4	810000	19440000
Travel	7	188160	948326400

Table 1.3: Characteristics of several negotiation domains

	AgentFSEGA	Agent K	Nozomi	AnAgent
Itex-Cypress buyer	0.671 / 0.721	0.653 / 0.703	0.548 / 0.732	0.576 / 0.776
Itex-Cypress seller	0.721 / 0.671	0.670 / 0.721	0.588 / 0.759	0.588 / 0.759
The small engineered domain - buyer	0.822 / 0.787	0.700 / 0.871	0.611 / 0.924	0.586 / 0.946
The small engineered domain - seller	0.787 / 0.822	0.664 / 0.935	0.664 / 0.935	0.596 / 0.955
SON buyer	0.773 / 0.773	0.882 / 0.587	0.877 / 0.612	0.972 / 0.137
SON seller	0.773 / 0.773	0.847 / 0.622	0.847 / 0.622	0.965 / 0.175
Travel buyer	0.860 / 0.744	0.743 / 0.835	1 / 0.647	0.751 / 0.769
Travel seller	0.744 / 0.860	0.741 / 0.672	1 / 0.198	0.763 / 0.608

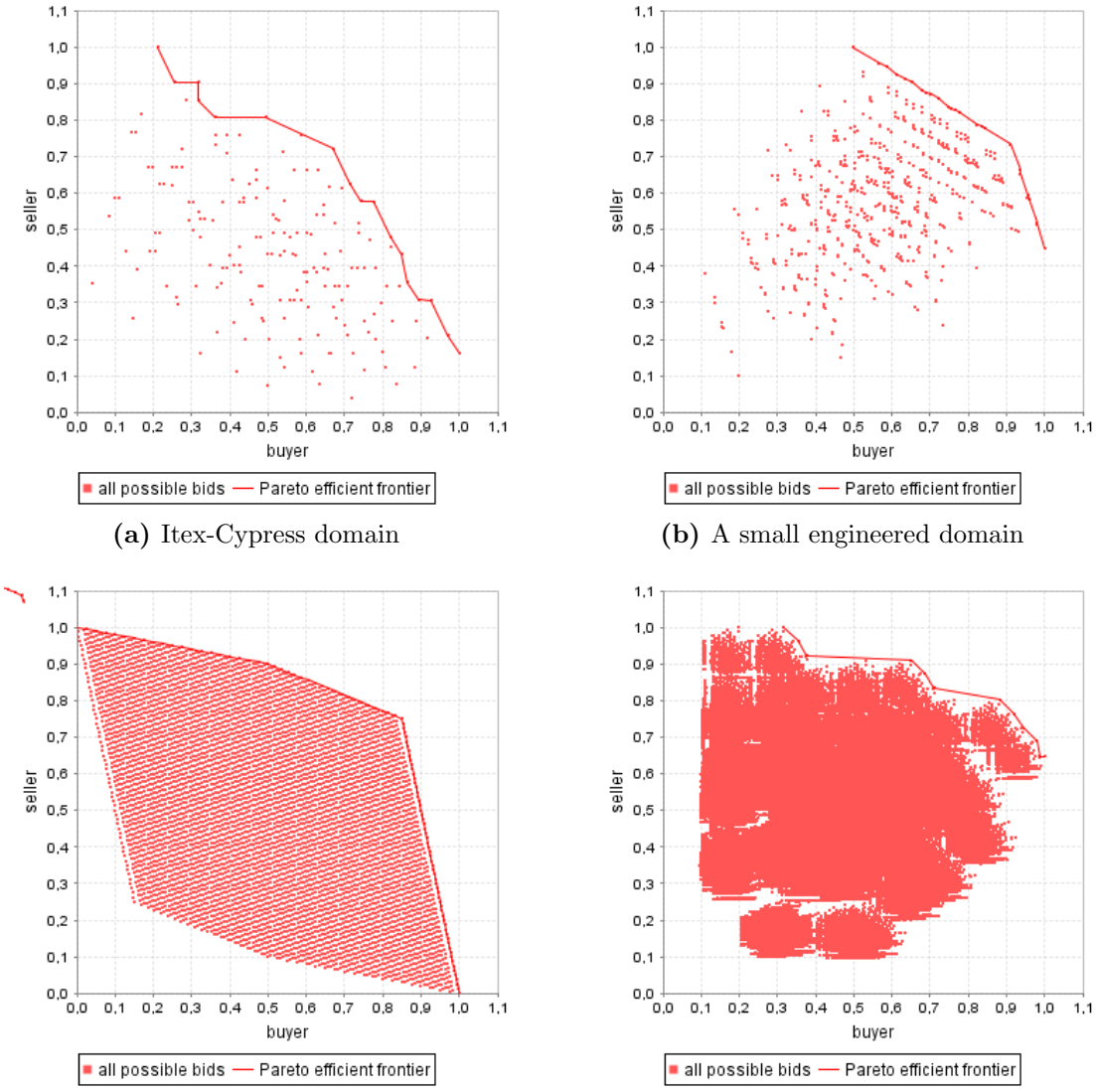
Table 1.4: AgentFSEGA against ANAC2010 agents. Columns represent the opponent agents. Each row represents a role played by AgentFSEGA.

consumer and the provider on each domain. We can notice that the consumer and the provider might have a similar profile - like in the SON domain (figure 1.35c) and in Itex-Cypress (figure 1.35a). Opposed, on the small engineered domain and travel, the buyer has fewer acceptable bids, compared with the seller. Thus, it is relevant to run experiments with AgentFSEGA playing successively both roles.

We tested AgentFSEGA against itself and against *Agent K*, *Nozomi* and *AnAgent*. All these latter agents originate from the Genius repository and participated in the ANAC 2010 competition[1]. The full logic of these agents can be found in [Ito et al., 2012] and they were tailored for time-constrained negotiations.

Table 1.4 presents the detailed results. On each domain, when playing against itself (the AgentFSEGA column), the outcome is an approximation of the Kalai-Smorodinsky solution for that particular negotiation game. We should note that in every game, AgentFSEGA scores a fair result. The bigger the negotiation domain, the better are the obtained results. This is because on bigger domains, the learning strategy is powerful and sensible enough to register the variations of the

[1]Agent K and Nozomi were implemented by the team at Nagoya Institute of Technology Japan and AnAgent by Bo Ann from the Univ. of Massachusetts Amherst USA



(a) Itex-Cypress domain

(b) A small engineered domain

(c) The SON domain

(d) Travel domain

Figure 1.35: SLA search cloud for several negotiating domains

opponent's profiles. Even if the domains and the role profiles are not symmetric (this is the case for the small engineered domain and travel), this fact does not affect the performance of AgentFSEGA. Figure 1.36 presents the outcomes of the negotiation games relative to the Kalai-Smorodinsky solution of that game and compared with the relative opponents' averaged score. On the small domains, AgentFSEGA is overpassed by the rest of the agents, but it succeeds to acquire

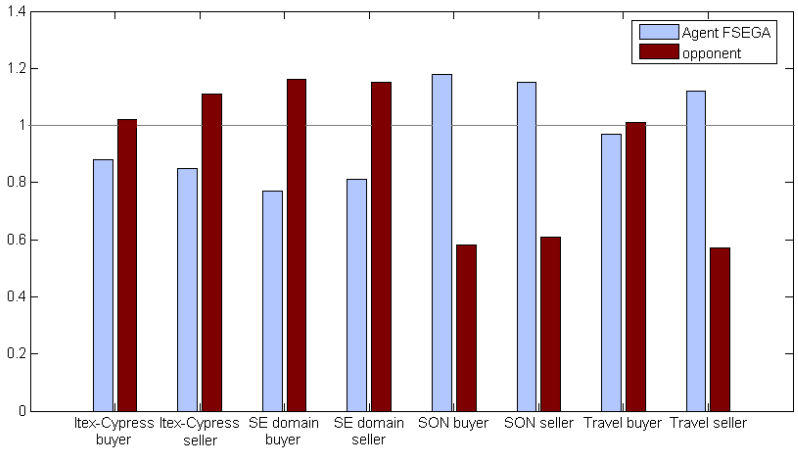


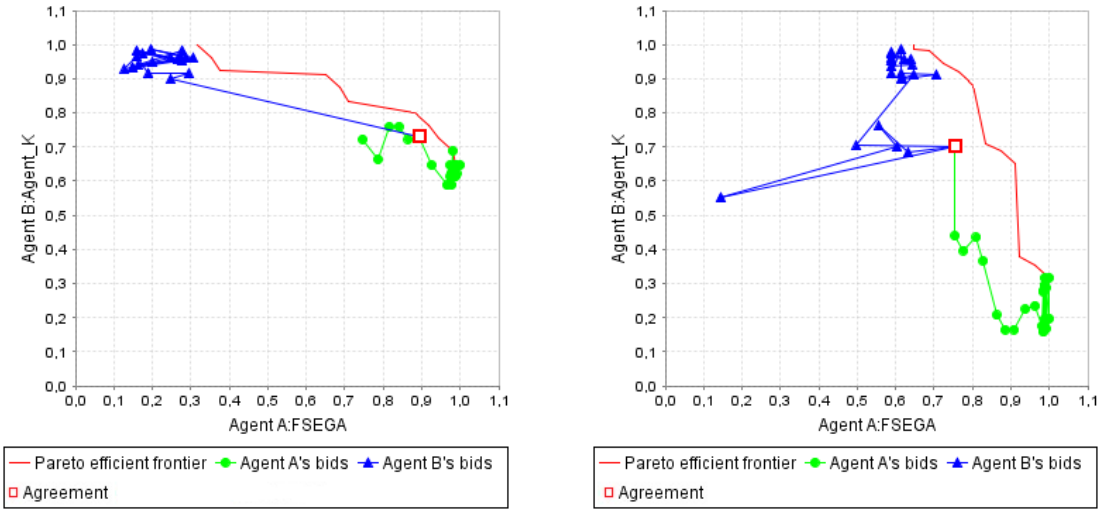
Figure 1.36: AgentFSEGA performance relative to the Kalai-Smorodinsky solution of a particular negotiation game.

about 80% from the target utility. On big domains, AgentFSEGA performs well, overpassing the other agents. Even on the travel domain the buyer's result is encouraging, because, as the reader can notice from figure 1.35d, on this domain the buyer has worse alternatives than the seller.

The travel domain is the most difficult, out of all domains we used for testing. The difficulty resides in the size of the domain, the big number of hypotheses and the fact that the domain is not symmetrical - one player has better alternatives than the opponent (see figure 1.35d). From figure 1.35d we can also notice that the Pareto frontier is a bit far away from the cloud of the possible agreements. In figure 1.37 we depict the performance of the FSEGA agent against itself on this domain. We can observe that the agreement is at the frontier of the outcomes cloud, on the line that defines the Kalai-Smorodinsky solution. Experimenting on the travel domain against Agent K (the most powerful agent from the Genius repository), we observe (see figure 1.38) that AgentFSEGA tries to move the agreement closer to the optimal possible allocation (the one obtained in figure 1.37).

Out of the experimentation presented above, we can draw two essential conclusions. First, if the intelligence employed during negotiation is the same on both parties, the outcome is approximating the Kalai-Smorodinsky solution, with all desired properties presented in section 1.5.2.4. Thus, such a sort of intelligence

Figure 1.37: AgentFSEGA against itself on the travel domain.



(a) AgentFSEGA as buyer

(b) AgentFSEGA as seller

Figure 1.38: AgentFSEGA against Agent K itself on the travel domain

can be a good alternative in open service-based architectures.

Second, if the AgentFSEGA plays against others, it scores fair results. Therefore, if some malicious user would like to undermine an open service-based system by replacing the designed intelligence for SLA negotiation, still good SLAs will result out of the negotiation. In this way, even in the presence of exploiters,

the SLA-based open system still succeeds to achieve its goals and continues to function. The allocation of the system resources is not disproportionate between the players. Of-course, here we do not claim that our agent is the most intelligent and there is no better solution, but we prove that with a time-constrained learning agent, we succeed to accomplish one of the important design goals of a SLA-based open system.

1.5.5 Conclusions and future work

This contribution tackles the automatic SLA negotiation in computational grids with competing resource owners and consumers. While previous work on SLA negotiation in grids focuses on providing negotiation models in the bargaining game, in this paper we introduce a framework for building automatic negotiation strategies under time constraints. We build our work on the latest results concerning negotiation strategies developed in agent research and specifically, on the Bayesian learning agent. To instantiate our general framework for intelligent negotiation strategies under time constraints, we further extend and modify the Bayesian agent.

We show that the time-constrained negotiation framework can help in building much robust and better negotiation strategies for service level values establishment, given that time is a key resource. By presenting the implementation of the Bayesian agent in the proposed framework, we present a valuable experience about how to extend an opponent modeling-based strategy to cope with the time restrictions.

In general, we show that an opponent learning-based negotiation strategy is a valuable asset in SLA-based open grid systems. Empowering the system with this type of intelligence can result in a equalitarian allocation and fair satisfaction of participants. Further, such an intelligent agent can become a mean towards the system dependability, in terms of stability and smooth functioning.

As a further work, other intelligent strategies need to be adapted in the presented framework and further testing is required, even on more complex and difficult negotiation domains. We still lack a formal proof that the Bayesian learning strategy converges towards the desired (optimal) SLA outcome.

1.6 Conclusion

In this thesis, we presented our scientific contributions developed after the PhD thesis defense. We approached the hot topic of dependable P2P systems and resource management in heterogeneous environments, contributing with various results towards the goal of automated collaboration between autonomous system participants. While during our PhD, we formally approached collaboration in multi-agent systems, in this thesis, we presents three major results that can enhance collaboration in open P2P systems.

This thesis starts with a short introduction to P2P systems. While P2P systems moves away from the initial file distribution goal towards service-based P2P systems, we introduced concepts like Service Level Agreements, structuring P2P systems, economics of P2P systems and the sabotage problem and desktop grids as a specific sort of P2P system delivering computing power. Related with all areas enumerated above, we presented some relevant scientific contributions which represent the state-of-the-art in the field.

Next, we introduced our main scientific contributions. First, we presented the topic of reputation management in computational grids. We did a full discussion on the trust and reputation concepts and reviewed other reputation models created for grid and P2P environments. Our reputation model is formally described within the SOA-based requirements, around the quality of service concept. The formal description of the model, depicted in the Z notation, employs the concept of utility. Rather than assessing peers based on the subjective testimonials of third parties, our model uses the monitoring services widely available in grid and P2P environments to collect traces about peers behavior. The reputation model is validated by experimentation with SimGrid and two usage patterns are proposed: (i) to assess the provider reliability in scheduling operations and (ii) to supply a user access model to grid resources.

The second major contribution tackles the problem of sabotage in P2P-enhanced desktop grids. While desktop grid nodes are linked by a P2P overlay and can communicate between them, they can exhibit colluding behaviors that might undermine the desktop grid. We defined five strong colluding behaviors and propose models to tackle them. Our statistical classification toolbox allows to distinguish with certainty, which are the honest peers in the desktop grid - in the sense that

they do not enter in collusion with others. Therefore, the remaining peers might be colluding ones and a cheap auditing can clearly decide which are or not the bad nodes. The solidity of our algorithm was in-depth demonstrated by experimentation, considering the full range of population structures and realistic desktop grid assumptions. We helped in the elaboration of the Maximum Independent Set approach for characterization of the peers, that further enables to tackle the whitewashers.

Our third contribution regards the strategic behavior of peers when negotiating about service delivery terms in bilateral interactions. Given the negotiation setup described in WS-Agreement Negotiation standard, we proposed a framework for constructing strategic behaviors, given that limited time is allowed for the bilateral negotiation process. All we require is that the peer to use some opponent-learning tactic. We envisage that the node should keep strong with his offers at the beginning of the negotiation time, in order to let the opponent learning scheme to converge and to have a better understanding about the opponent profile. Only when the negotiation time is about to elapse, our agents are making concessions, in order to achieve a reasonable deal. With our framework, we constructed a SLA bidding strategy starting from the baseline Bayesian agent. With this strategy, we played many negotiation games against various other agents proposed by the literature and we showed that, in general, we obtain good SLAs and we keep a fair division of resources within the society.

With this thesis, we showed that concepts, tools and models extracted from economic theory are worth for consideration in distributed computing, in order to design automated systems with open participation and enhancing loose collaboration between participants. As we presented in our future research prospects, economic theory might help to design the business models of nowadays computing service providers, given that resources originate in private and public clouds, grids and desktop grids.

Chapter 2

Career Development Plan

2.1 Teaching directions

In this section, I presents the on-going and future directions in what concern my teaching activity. My teaching activity is subsumed to the fact that I hold the chair of Business Information Systems (BIS) at Babeş-Bolyai University (BBU) of Cluj-Napoca and I'm the study director of the Bachelor program in Business Information Systems at BBU.

According with The Joint Task Force for Computing Curricula 2005 [2005], Information Systems (IS) represents one of the main directions for computing-oriented study programs in USA and internationally, together with computer engineering, computer science, information technology and software engineering. Specialists on IS focuses on integrating IT solutions and business processes to meet information needs of businesses and other enterprises. It emphasizes on information and its key role in the enterprise, around the systems and technologies that generate, process and transmit information.

The Computing Curricula 2005 referred above, clearly position the IS degree programs in correspondence with the other sorts of computing degrees, recommending the suitable disciplines that should be part of a BIS curricula and also, what competencies and abilities a BIS degree graduate should possess. In accordance with these world-wide adopted recommendations, considering the specificity of the Romanian software market and business environment, together with my colleagues in the BIS department at Babeş-Bolyai University under my direct

coordination, during spring 2010 we revised the study plan for the BIS bachelor degree offered by BBU.

Consulting the ACM/AIS Model Curriculum and Guidelines for Graduate degree programs in IS [Gorgone et al., 2006], following a research about the disciplines delivered by European universities in IS master programs, in parallel with the revision at the Bachelor level, we revised the study plans for the two master programs offered by the BIS department at BBU: EBusiness and Decision Support Systems for Business.

All three programs mentioned above were assessed by the Romanian Agency for Quality Assurance in Higher Education in June 2011, obtaining its credential with the highest possible degree.

This section evolves as follows: subsection 2.1.1 presents the three study programs running part of the BIS department at BBU. I will emphasize on the flows of the study topics, the competencies we give to the graduates. This section briefs the achievements and the improvements we introduced in the last two years, under my direction. Next, in subsection 2.1.2 I describe the items I see mandatory for the evolution of those three degree programs in the following years.

2.1.1 Status Quo

Our study program in BIS resulted out of the national consultations that took place between 2005-2006, together with the adoption of the Bologna process in Romania. In autumn 2009, further consultation at national level happened in order to unify the competencies delivered by the Romanian Bachelor study programs in Information Systems. In 2009, I took part of these consultations, as study director and local chair of BIS.

This last consultation decided that Romanian study programs in IS should deliver their students with the following core competencies:

- Usage of concepts, theories, and research methods for business processes
- Efficient usage of computer systems, operating systems and Internet, including the team development of efficient solutions for management and configuration of operating, communication and computer systems

Master	**E-Business** – oriented on web technologies and distributed systems			
	Decision support systems for business – oriented on information systems and intelligent methods for enterprises			
Sem. 6	Software engineering (UML)	Internet in business (virtual stores, Ajax)	Advanced databases (Oracle)	ERP products Artificial intelligence
Sem. 5	Object-oriented programming (Java) RAD (.NET)	Web sites design (PHP)	Databases in business (fundamentals of DB, SQL)	Testing
Sem. 4	Algorithms and data structures Introduction to programming (C)	Graphics (web design, Java Script)		Computer networks Operating systems
	Programming / software design	**Web technologies**	**Databases**	**Enterprise and other disciplines**
Business subjects (sem. 1 to 3)	Micro-macro economics, mathematics, fundamentals of accounting, management, and marketing, finance, financial accounting, statistics		IT databases for office usage (MS Office, FoxPro and Access)	

Figure 2.1: The study plan for the Bachelor program in Business Information Systems at BBU

- Professional usage office software for business purposes, including the design of a flow-oriented business process for data processing
- Development of software components using data structures, algorithms, programming techniques and advanced programming languages
- Development of software applications using databases, multimedia resources and client-server technologies
- Maintenance of information and computer systems, including business analysis and requirements gathering for information systems, system design and software process management

Figure 2.1 presents the structure of the study plan for the BIS Bachelor program, as evaluated by ARACIS on June 2011 visit. ARACIS acknowledges that the BIS study program presented below fulfills the core competencies presented above and represents a high quality BIS program in Romania.

According with the Bologna principles, the Bachelor study program is organized in 6 semesters. The first three semesters are common with other business programs offered by the Faculty of Economics and Business Administration at BBU. Students get the basic business topics like fundamentals of management, marketing and accounting. They learn how to administrate a company with subjects like finance and financial accounting. Besides these business fundamentals, they also learn core economics, including micro and macro economics, fundamentals of mathematics applied to economics, actuarial mathematics and descriptive statistics. During these introductory semesters, students get only flavors for IT for business, learning tools like MS Office (with emphasize on Word, Excel and PowerPoint), MS Access and FoxPro, being able to handy operate with spreadsheet processors and databases. After these three semesters, students get a core business-oriented competencies, as advised by The Joint Task Force for Computing Curricula 2005 [2005].

The BIS program really starts only in semester 4. From this point up to the end of the study plan (semester 6), students get computing topics, grouped in four main pillars: (i) software programming and design, (ii) databases, (iii) web technologies and (iv) IT and enterprise disciplines. The subjects and technologies for the courses covered in these last three semesters are selected such as the students to be able to compete in the jobs market with graduates from theoretical computer science and computer engineering.

On the software programming and design pillar, students get the fundamentals of programming and algorithms and data structures subjects, both delivered with the help of the C programming language. Next, in the 5th semester, students learn the object-oriented programming principles in Java and a first Rapid Application Development tool (Microsoft .NET). Despite the fact that western IS programs does not offer a clear development tool-oriented course, like .NET, we opted to introduce this subject to our students, given the strong requirement of the job market for .NET programmers. After the completion of the 5th semester, students are able to design and implement small programs in various languages (C, Java and .NET), including web programming in PHP. Therefore, in the last semester, they can learn how to organize and manage larger software projects, by having a first topic of software engineering.

Part of the databases pillar, students learn the fundamentals of database sys-

tems, including structuring a database, the normal forms and relational database management systems. The course delivered in the 5th semester also covers relational algebra and QBE and introduction to SQL. The second advanced course in databases (6th semester) covers distributed database structuring and working with the Oracle DB server. Students deeply learn PL/SQL.

The web technologies pillar grows from the basic computer graphics course of semester 4, which introduce the students to front-end web programming (including cascading styles and front-end scripting). During the 5th semester, students learn back-end web programming with PHP, getting the grasp of a widely used web technology. The 6th semester course asks the students to design and develop larger web projects, including virtual stores, moving them into the world of E-Commerce technologies.

Besides these 3 pillars, where the students get knowledge and competencies highly asked by the local job market, the study plan includes other mandatory subjects for computing degree graduate: computer networks, operating systems, fundamentals of artificial intelligence, software testing - there is a large demand on the local market for testing engineers and ERP systems operation.

The study plan presented above get crystallized in during the consultations that took place between 2005-2006, with the adoption of Bologna process in Romania. With the revision of spring 2010, we introduced the second programming class in the 4th semester - responding the ask of the students to give them double time for learning the basic programming, we changed the focus of the object-oriented programming class from C++ to Java, we introduced the world-wide adopted study materials for disciplines like Software Engineering and Artificial Intelligence.

At the master level, each Romanian university has a higher autonomy to decide about the clear specialization of a master study program and the specificity of that program. The BIS department at BBU decided to deliver two master programs: one oriented on web technologies - named EBusiness, and one oriented on enterprises - named Decision Support Systems for Business (SADE). Figure 2.2 presents the study plans of these two master programs. Initially, these master programs were devised together with the Bachelor program in BIS. The study plans described in figure 2.2 emerged after the revision that took place on Spring

The PhD program in BIS			
Sem 4. (joint)	Preparing the dissertation 2 elective subjects out of: Software quality assurance, Elearning, Game theory, Cloud computing, Social networks, Websites optimization, Human computer interaction		
Sem. 3 EB	M-Business EBusiness programming and design Virtual business		Security in EBusiness EBusiness technologies and applications
Sem. 3 SADE	Business Intelligence Programming with components	Expert systems Advanced ERPs	DSSs
Sem. 2 (joint)	Financial management Communication skills	Management of software projects Advanced software engineering	Computational intelligent methods Distributed systems
Sem. 1 (joint)	Business governance Inferential statistics	Advanced programming (.NET)	Knowledge management
	Business subjects	**Information systems design and development**	**Technologies**
Bachelor	Business Information Systems Other business and computing degrees		

Figure 2.2: The study plans for the two master programs in Business Information Systems at BBU

2010. That revision includes the outcomes of a study including the ACM/AIS guidelines for master programs in IS [Gorgone et al., 2006] and disciplines offered by western universities in similar study programs. The ARACIS visit of June 2011 evaluated these two programs and gave them full support and the biggest degree of confidence.

For these two master programs, we recommend our Bachelor graduates in BIS to continue at the master level, in order to complete and strengthen their IS background. The fulfillment of the BIS Bachelor degree is not a mandatory entry requirement, the master programs being open to both other business and computing programs graduates. The first study year is common for the two master programs, consisting of disciplines with mandatory technological and business background. The 3rd semester represents the specialization of each master program. The 4th semester lets the students to decide about elective classes and to prepare their dissertation.

The first joint specialization study year consists of the disciplines grouped in three main categories: (i) business subjects, (ii) information systems design and

development and (iii) technologies.

To be prepared for the business environment, students should possess strong communication skills offered by a course of language inter-cultural communication in business. Being prepared to overtake decision level position in business, students should get competencies in business governance and financial management. Inference skills are also mandatory, therefore, the study plan includes inferential statistics.

The information systems design and development pillar represents the core specialization of a master degree in IS [Gorgone et al., 2006]. Therefore, students take an advanced programming class and two classes of advanced software engineering and management of software projects.

The technologies pillar is also of major importance. Given the nowadays move of computing towards virtualization and networked applications, students should take the basics of distributed systems and parallel programming. Knowledge management is mandatory for both web-based or enterprise information systems. A core course in Computational Intelligent methods allow the students to get the fundamentals of datamining and evolutionary computing.

The EBusiness program is specialized for web-oriented information systems. Therefore, students get a class of Mobile Business, one of EBusiness design and development covering Business Process Management, Security in EBusiness, EBusiness technologies and applications and virtual business.

The SADE program is specialized for enterprise information systems. Students take modern classes of Business Intelligence - learning the Oracle Business Intelligence solution, Expert systems, Decision Support Systems, advanced ERPs - where they learn the fundamentals of the SAP technology and programming with components - enabling students to devise complex software applications.

The 4th semester let each student to choose his study path. We offer them electives from a wide range of topics, like Cloud computing, software quality assurance, human computer interaction - to enable students for professional GUI design, social networks and websites optimization - for a stronger web orientation, and game theory and ELearning as niche classes. Besides following two out of the above enumerated classes, students should prepare their dissertation and strongly interact with the dissertation advisor.

The revision of the study programs in spring 2010 - which I coordinated,

consisted on the following additions:

- Introduction of inferential statistics, instead of marketing specialized class. We considered that inferential statistics better fits the computing orientation of a IS master graduate
- Introduction of the topics of distributed systems and computational intelligent methods. We considered that these topics are mandatory, given the current advances in computing, towards world-wide information systems, virtualization and multi-core enabled hardware. I particularly contributed in devising new study materials for the class of Computational Intelligent Methods and some lectures for the Distributed Systems class
- The knowledge management class started to contain web-based knowledge management topics and uses tools from Semantic Web
- Introduction of the Business Intelligence and Expert systems classes for SADE
- Introduction of the Security in EBusiness and Technologies and applications in EBusiness for the EBusiness program
- Changed the 4th semester elective classes, by adding classes of software quality, cloud computing, HCI, social networks and websites optimization. We also enabled all students to select from all available topics, giving them freedom to select their most appropriate elective.

Students acknowledged the Spring 2010 revisions of the IS study programs. Now, both the Bachelor and the master study programs entered their second generation of students. Particularly, the 3rd year BIS students and our master students registered during 2009-2010 were consulted during the revisions of the study plans.

During summer 2010, after the new study plans were released to all students of the faculty, the BIS study program was selected by 137 students, currently, having them registered on the 3rd year of the Bachelor BIS program. This is the biggest figure of students the BIS degree at BBU ever recorded. On summer 2011, more than 100 2nd year students of the faculty selected the BIS program. And,

on the same summer, during the admission process to the university programs, we recorded a demand of more than 200 students to be registered in the 1st year of study. We were able to a

The master programs attracted on average a total of about 50 students / academic year. On the academic year 2011-2012, we recorded 44 students registered to the EBusiness program and 19 to SADE.

2.1.2 Didactic proposals

In this subsection, I will present several directions to further improve the above presented study programs. I will present my opinions about future improvements to the mentioned study programs, according with two world-wide initiatives.

First, we mention that ACM and AIS joint task force produced in May 2010 a new IS undergraduate curriculum guidelines [Topi et al., 2010]. These guidelines acknowledges the challenges in front of the information systems development process: the globalization of the software development process, ubiquitous use of web technologies, the emergence of new architectural elements including web services, software-as-a-service and cloud computing, integrated ERP systems, pervasive mobile computing and governance models applied to best practices in IT.

Second, we mention the NSF/IEEE-TCPP Curriculum initiative on Parallel and Distributed Computing - Core Topics for Undergraduates[1]. The IEEE Technical Committee on Parallel Processing established this initiative to setup basic guidelines of parallel programming and distributed computing items adoption as early as possible by various computing-oriented undergraduate programs. We participated in the first workshop of this initiative that took place part of IPDPS 2011 conference and in-line with the findings and the challenges recognized by the IS2010 document [Topi et al., 2010], we consider that is worth of consideration of various such guidelines.

Curricula orientation towards facilitating students early entering the jobs market

In order to facilitate our students entering the jobs market, our degrees need to deliver those competencies, skills and abilities required by the companies, es-

[1]http://www.cs.gsu.edu/ tcpp/curriculum/?q=home

pecially by those located in our vicinity. Thus, a strong collaboration between us and the local companies would be required. We target for our students jobs like business analyst, software analyst and consultant, software developer, software engineer, software/system architect, project manager, web developer, web designer and web engineer, database expert, quality assurance expert, or IT expert. Local software houses operating around Cluj-Napoca represent the most wanted companies for our students, but they can find good placements in companies operating in other business sectors. Regularly, we meet representatives of the companies in order to adapt our curricula. The introduction of the *Software testing* topic in the Bachelor curricula came out of such a consultation.

Below, we present our focused proposals with this respect.

- at the Bachelor level, teach the students those fundamental topics of computer science that are mandatory for a good software engineer, like basic programming, object-oriented programming, fundamentals of databases and fundamentals of software engineering
- exemplify the fundamental computer science concepts with up-to-date software tools and languages. E.g. we envision the shift on teaching object-oriented programming from C++ to Java or usage of modern tools for the Software engineering
- in terms of programming languages and software technologies, try to cover as much as possible. We teach C, Java, .NET, PHP and Oracle.
- at master level on the 3rd semester, ask companies to deliver specific course modules of their interest. Many local companies attract their employees by offering them courses organized in-house. With this proposal, we ask companies to organize their lectures part of our curricula, directly targeting our students with specific topics. We experienced good collaboration with MSG Systems[1] - who delivers a SAP course, and iQuest International[2] - who deliver a Business Intelligence module. Both collaborations are part of the DSS for Business MSc program.

[1] http://www.msg-systems.ro/
[2] http://www.iquestint.com/

- propose companies to host master students during the 4th semester in order to help them to elaborate their dissertation. Realization of the dissertation inside a company is common for western universities, while in Romania, this practice is only at the beginning. How it works: the company propose a dissertation topic to the student; the student - together with the advisor from the university, agrees on the topic; and the student works on the topic and elaborate the dissertation inside the selected company. For such a joint dissertation with companies, a student works in the company for at least three months and has a better chance to see and learn the industry working style. The companies have the advantage to target the best students before their graduation. In general, good joint work is produced out of such joint projects, but a student doing the dissertation like this, in general, will put a higher effort in his work comparing with the case he works alone for the dissertation.

Introduction of modern topics changing the information systems today

During history, every computing era was characterized by some major technological innovations that shifted the whole science and industry. Today, we face the transition towards computers equipped with multi-core processors. Thus, as acknowledged by various institutes, including the IEEE and NFS, students need to be faced with parallel and distributed technologies as early as possible.

We propose the following punctual directions to be adopted in our curricula:

- teach students the core parallel architectures and basic low-level programming as soon as possible. The *Operating Systems* course should be adapted to the novel multi-core technologies and should exemplify the organization and architecture of the multi-tasking operating systems. *Introduction to programming* course should include a module dedicated to low-level programming with pointers
- the *Algorithms and Data Structures* course should exemplify the core algorithms also with their parallel implementations

- the *Rapid Application Development* course should include a module about multi-threading
- the *Advanced Databases* course should contain a module about distributed databases
- at the master level, the *Distributed Systems* course should introduce the main parallel programming models and present architectural issues about distributed systems organization
- at the master level, the *Emerging web technologies* course and *Cloud computing* should introduce the students the facilities available from virtualization and de-localized computing.

Keep a strong focus on the specific items of Information Systems management and development

Information Systems represent the core subject of our degrees. Thus, beside everything else, we need to keep a strong focus on the specific subjects of Information Systems. Our punctual proposals are as following:

- reorganize the Software engineering disciplines in order to teach students both the software process - including the management of the software production life cycle, and the software engineering - including the techniques about how to build robust information systems
- provide the students with a specific discipline tackling *Requirements Gathering*. As we see our students as business analysts and software consultants, they need to know how to formally interact with the business people for analysis and gathering the specific requirements of an information system
- provide students with a specific discipline tackling organization of information processes within organization. We refer here to *Business Process Management* - engineering and re-engineering information workflows within organizations
- give students abilities about how to deploy, operate and manage IT or software instruments within organizations and let them during the studies to experience with systems like ERPs, EASs or others

2.2 Research directions

In this section I will present a research proposal for the 3 years, intended for a small group of 5 researchers: one principal investigator, two postdoc researchers and two phd students. Before presenting the research proposal, I will shortly introduce the research group on automated collaborative systems, which I established part of the Business Information Systems department at Babes-Bolyai University.

2.2.1 Research group on automated collaborative systems

In 2008, Gheorghe Cosmin Silaghi established the research team focusing on automated collaborative systems, part of the Business Information Systems department. Part of this team, several PhD students - Mircea Moca, Cristina Stefanache and Ioan Petri, graduated their theses and published valuable contributions. Currently, the team hosts another 2 PhD students - Gabriela Morar and Cristina Muntean, scheduled to finish their PhDs on September 2012 and several master students worked as researchers (Liviu Dan Serban - graduate student of class 2011, Alexandru Butoi and Andreea Ilea graduate students of class 2012). The experienced researchers part of the team are G.C. Silaghi, Cristian Marius Litan (PhD in Economics - game theory from Carlos III University Madrid), Mircea Moca and Prof. Dr. Nicolae Tomai - PhD advisor in Romania. This research team represents the foundation on which G.C. Silaghi started to be involved in the FP7 security research projects mentioned in the CV. The main research idea of the team is to investigate and employ fundamental research results (mainly originating from artificial intelligence and economic theory) in developing information systems of practical usage, especially from the field of distributed systems.

The research group led by G.C. Silaghi is involved in collaborations with groups from abroad, including Prof. Omer F. Rana from Cardiff University, Gilles Fedak from INRIA Lyon France, Prof. Catholijn Jonker from Delft University of Technology, Radu Prodan from Innsbruck University and others.

The long term objective of G.C. Silaghi is to establish and run - based on the core researchers named above, the Research Center in Business Information Systems at Babes-Bolyai University with independent funding, based on the projects won by its members.

2.2.2 Efficient resource management in heterogeneous clouds

2.2.2.1 Problems

Nowadays, computing undergo a big movement towards virtualization and usage of cloud resources on pay-as-you-go basis [Buyya et al., 2008], while each company and department carefully decides about acquisition of new hardware and software and the increased utilization of the existing computing infrastructure represents one major business objective. All this movement happens in the context of modern enterprises delivering high technology added-value products, with growing demand for high performance computing infrastructures. Commercial cloud computing enables the separation between a software service owner responsible for updating and managing a software capability encapsulated as a service and an infrastructure provider, primarily offering computational, data and network resources that may be used to deploy the software service. When offering the cloud services as infrastructures, the end user gets a virtual machine equipped with some software stack and then, it can control the operating system of the machine and run the installed software. In this research proposal, we will tackle the business of a Software-as-a-Service (SaaS) provider, who intends to optimally deliver its software capability, while not owning enough infrastructures to meet all its customers demand. The problem is relevant for the nowadays computing, with the emergence of IaaS providers like Amazon Elastic Compute Cloud - delivering virtual machines on request, with the emergence of virtual appliances marketplaces - like the VMWare VA marketplace - where one can find its requested software appliance ready to be deployed on a new infrastructure and multi-tenancy, where a single instance of a running software serves multiple tenants.

We can shortly formalize the addressed problem as following. A service owner acts as a SaaS provider, delivering, on demand, some services to his clients. Service delivery between the client and the service owner is regulated by a service level agreement (SLA), mentioning the quality of service (QoS) properties, the price paid and the penalties incurred by the provider for violating the SLA. For service delivery, the SaaS provider owner uses a heterogeneous cloud, being able to launch new tasks on existing computing nodes or to deploy new virtual machines over various infrastructures, including private clusters with limited resource

availability, IaaS providers like Amazon EC, volunteer computing resources like BOINC-based projects or other cycle stealing HPC systems like Condor or other sorts of resources. We assume that the service owner sees all these heterogeneous (cloud) resources as a large-scale data center comprising n heterogeneous computing nodes, grouped according with their location or their provider. Each computing node has a CPU with performance defined in MIPS, RAM and external associated storage with various capacities and some network bandwidth for data transfer. During our research activity, we will identify other properties of the computing nodes that might become relevant for our study. During time, customers submit requests with various service level needs to the SaaS provider and this should deploy virtual machines over the available computing nodes or should instantiate new computing resources in order to fulfill the user requests.

We will investigate and define the business models and the options available for SaaS provider for his resource management, in order to achieve one or several desired efficiency properties, as follows: (i) total profit maximization, (ii) profit maximization per customer request, (iii) increased customer satisfaction - meaning that the provider would target to serve as many customer requests as possible, (iv) reliability - meaning that the provider would target to successfully serve the accepted requests, (v) energy efficiency. Other efficiency metrics could also be defined and investigated. We consider that the service owner business evolves in a computing environment characterized by varying customers demand, various pricing policies developed by the owners of some heterogeneous infrastructures or costs (including the ones for the supply of the non-functional customer requirements) incurred by other infrastructures, including the networking and new infrastructure deploying costs. To achieve efficiency - related with one or more of the properties enumerated above, the service owner might take tasks scheduling decisions, consider resource allocation policies, solve the trade-offs between cost and energy consumption and performance, might consider virtual machine and data migration policies between computing nodes, computation and data replication in the attempt to increase service delivery reliability and minimize SLA violations. To analyze and solve the problems mentioned above, we will adopt an agent-based approach, depicted in fig. 2.3.

Three sorts of agents are relevant in our approach: the *computing node agent* (CNAg) - managing the computing nodes, the *infrastructure owner agent* (IOAg)

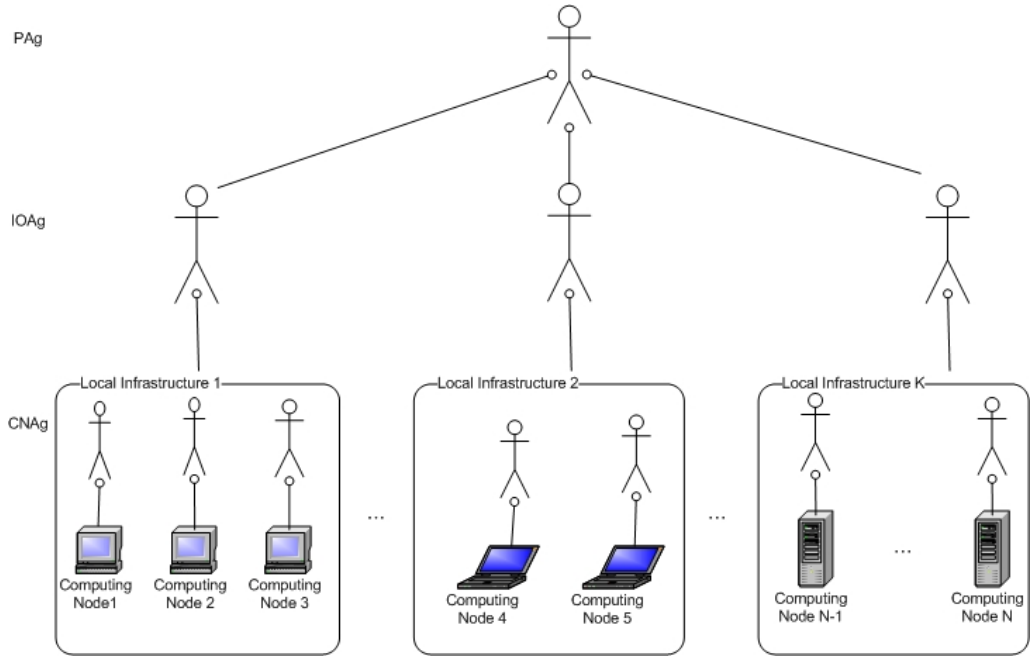


Figure 2.3: An agent system deployed for managing a heterogeneous cloud infrastructure

- managing all nodes deployed on a given infrastructure and the *provider agent* (PAg) - responsible to dealing with the customer requests and deciding about some resource allocation per request. We devised this three layered architecture with autonomous agents in order to allow decision decentralization and assure self-healing and self-organization properties to the overall system, permitting bottom-level agents to decide on various problems, transparent of their upper level coordinator. Thus (e.g.), CNAg residing on the same infrastructure, might decide to migrate VMs or tasks between their computing nodes in order to minimize energy consumption or avoid overheating, IOAg can compete between them to attract tasks on their managed infrastructure from the PAg etc. With this approach, we will need to define interaction protocols and behaviors for these sorts of agents and we can investigate the effect of some policies (like resource pricing policy, replication policy etc.) over the performance of the service owner. We also allow gradual introduction of novel infrastructure types to the system, or of novel infrastructure providers.

Below, we present the state-of-the-art with respect to the problem described

above. We position our research proposal with respect to this state of the art.

Only recently, after cloud adoption, literature started to tackle problems like the ones mentioned above. Litoiu and Litoiu [2010] take position towards optimal resource management within the SLA regulated generic SOA model. They define the cloud cost and penalties for SLA violation and the mathematical equation that should be optimized for profit maximization, given some constraints. Beloglazov and Buyya [2010] position themselves towards energy efficient resource management in virtualized cloud data centers, identifying some research challenges with this respect and providing some simulation results with respect to some allocation policies. Thermal management as suggested by Sharma et al. [2005] can bring further energy savings. Sedaghat et al. [2011] enumerate some problems faced by a cloud infrastructure provider, surveying various contributions related with cloud management, including the adoption of the business concern by low level managers, solving the optimization problems for satisfying the service business objectives, and autonomic management solutions. They formulate an overall unified governance model towards commercial management of a cloud infrastructure provider, but their model is not validated and is targeted towards IaaS providers. A Service Level Agreements (SLA) is a contractual obligation between a provider and a consumer defining the mutually agreed expectations [Andrieux et al., 2007] and it represents the key element towards a business ready infrastructure empowering the service economy in a flexible and dependable way [SLA@SOI Consortium, 2008]. In the context of cloud computing and the SaaS delivery model, as indicated by the RESERVOIR model [Rochwerger et al., 2009], SLA management represents the key element, software components being responsible to dynamically adjusting the committed resources for service level objectives fulfillment and reducing the number of SLA violations. SLA management will stay in the center of our approach, as indicated by the literature.

Recent papers propose solutions to achieve various efficiency properties. Task scheduling heuristics are among the most studied. Our interests go for studies developing heuristics for commercial setups, where providers are maximizing their profit, meantime fitting the agreed SLAs, or maximizing the delivery performance within some budgetary constraint. Salehi and Buyya [2010] propose two scheduling policies based on cost and time optimization for a service provider who needs to increase its local resources by hiring additional infrastructure from

Amazon EC2. Silva et al. [2008] propose a heuristic to optimize the number of VM to be allocated on a particular infrastructure for a bag-of-tasks, with budgetary constraints and for a maximal speedup. Paton et al. [2009] optimize the utility function based on response time and number of QoS target met in order to program an autonomic workload mapper that assigns tasks on various execution sites. Moon et al. [2010] use cost-based scheduling of cloud resources for profit optimization. Wu et al. [2011] define a mathematical model for the profit and minimize it, while reducing the number of SLA violations, when considering the mapping of tasks to virtual machines hosted by own cloud resources or a rented infrastructure.

Energy efficiency attracted several recent contributions, in-line with the proliferation of green computing. Goiri et al. [2012] propose a scheduling technique for energy efficiency in virtualized data centers, dealing with multiple facets restrictions, like economic modeling of the provider benefits. Their study is restricted to a virtualized data center and their heterogeneity refers to various capabilities of the computing nodes and various ownership of hosts is only briefly considered. Kliazovich et al. [2010] introduces GreenCloud a simulator that captures details of the energy consumed by data centers components, allowing simulations of resource management schemas for data centers components. Energy reduction can be obtained by task consolidation, and Lee and Zomaya [2012] propose heuristics for energy-conscious task consolidation, accounting both the active and idle energy consumption.

Autonomic management and self-organization is another issue approached when dealing with cloud resources management. IBM Tivoli Intelligent Orchestrator [Das et al., 2006] represents an agent-based solution for automated provisioning for Internet data centers, being capable to deploy and optimize resources at run-time in response to changing business objectives. Li et al. [2009] present an autonomic deployment solution for new applications on the CERAS cloud, given cost and QoS constraints, based on tracked performance model built from operational data for adjusting the deployment. Part of the FOSII project financed by the Vienna Science and Technology Fund, Maurer et al. [2010] implement a knowledge management system on cloud resources with help of advanced monitoring and autonomous control. Ranjan and Zhao [2011] propose a peer-to-peer approach for managing the services offered by a large scale, dynamic and evolving

cloud environment. The P2P approach with its routing and information dissemination particularities, avoids the bottlenecks of the centralized and hierarchical system approach.

We plan to integrate desktop grids and volunteer computing as part of the heterogeneous cloud resources. Their main property is the uncertainty, because one desktop grid node might disappear at any time, without returning the results for its assigned tasks. Thus, of risk management will become essential, as the SaaS provider should have warranties that its deployed tasks are solved, even in the volatile conditions of the desktop grid nodes. Building profitable systems on such resources is not a new idea; we mention here the study of Popovici and Wilkes [2005] which evaluate profit-based scheduling and admission control algorithms, addressing especially the resource-availability uncertainty. The study of Irwin et al. [2004] presents a value-based task scheduling, considering that a task yield decays linearly with its waiting time. They follow the metaphor that the heuristics balance the risk of future costs against the potential for gains in accepting and scheduling tasks. Risk analysis is performed commercial computing services [Yeo and Buyya, 2009], with the help of metrics like reliability, profitability, performance and volatility. For cloud systems, Fito et al. [2010] propose the SEBCRA toolkit for BLO-driven risk management, by assessing the impact of a violating a BLO, correlated with the probability of such event to happen. In general, on desktop grids replication is used to assure computation reliability and protection against sabotage [Silaghi et al., 2009]. On cloud computing marketplaces, trust management systems like the one devised by Habib et al. [2011] can help customers for selecting the appropriate provider.

Up-to-date, various providers delivered cloud software that allows one to build private clouds on-demand. We mention here only open source platforms like Open Nebula[1], Eucalyptus[2], OpenStack[3], Nimbus[4] or CloudStack[5]. Simulators like CloudSim[6] or MDCsim [Lim et al., 2009] allow researchers to seamless model, simulate and experiment emerging cloud infrastructures and services, before the

[1]http://opennebula.org/
[2]http://www.eucalyptus.com/
[3]http://openstack.org/
[4]http://www.nimbusproject.org/
[5]http://www.cloudstack.org/
[6]http://www.cloudbus.org/cloudsim/

real implementation of a design decision. CloudCast [Montresor and Abeni, 2011] presents a PeerSim[1] implementation of a message-diffusion protocol for extending a cloud environment with resources collected from P2P environments. Thus, PeerSim also represents an option for cloud simulation. Other solutions like EmotiveCloud[2] or Aneka[3] allow programmers to use various popular programming models like MapReduce, Task or Thread and seamless deploy applications in multiple infrastructures, including desktop grids. CometCloud[4] enables the deployment of real-world applications of hybrid federated infrastructures, including public or private clouds, grids or data centers. Hadoop[5] allows one to write programs for large datasets for distributed processing across thousands of machines.

In general, all these software are in their beginning, being not easy to deploy them or to work with their APIs. All of them need further developments and new models about how to distribute tasks on the infrastructures and self-organization could be devised.

2.2.2.2 Objectives

Our main objective is to study various alternatives available for a SaaS provider in order to efficiently deliver its services, making use of heterogeneous infrastructures. We will investigate and define the business model of a SaaS provider from various efficiency perspectives, including profit maximization, energy efficiency, risk management or customer satisfaction. We will consider at least the following infrastructures: public cloud providers, private clouds and desktop grids, all with their practical properties. The business model of the provider will include its pricing policy, the SLA establishment and delivery policy, the rules that manages the SaaS provider interaction with its infrastructure suppliers and the service deployment model. To optimize the SaaS provider business and deployment model, we will use a market-based approach, with a collection of agents deployed for each infrastructure component, and organized such as to emerge the good will out of

[1] http://peersim.sourceforge.net/
[2] http://www.emotivecloud.net/
[3] http://www.manjrasoft.com/products.html
[4] http://nsfcac.rutgers.edu/CometCloud/CometCloud/CometCloud_Home_Page.html
[5] http://hadoop.apache.org/

competing interests. Several sub-objectives are subsumed:

- To design the provider strategy for tasks and data placement on various infrastructures
- To design the task and VM migration strategy between the nodes of a local cloud infrastructure or from a IaaS provider to another
- Design the risk management policies, given the efficiency goal of the SaaS provider
- Show how the business model of the SaaS provider changes with the changes of the properties or cost models of the IaaS providers, or by considering a new alternative IaaS provider.

The agent-based solution employed for achieving the self-organization of the system will represent the main originality issue. The agent system will allow the realization of the efficiency properties. We will employ techniques from economics - including mechanism design and machine learning, in order to design the interaction protocols between agents and in order to predict and anticipate future customers' demand.

2.2.2.3 Impact

As cloud computing is an emerging field, our research will complement the existing cloud research with our novel agent-based approach. For the biggest impact, we envisage that our scientific results to be presented at top conferences including IPDPS, EUROPAR, CCGRID, IEEEE/ACM CLOUD and GRID and others and we will submit journal publications to top journals like Future Generation Computer Systems, Journal of Parallel and Distributed Computing, Concurrency and Computation: Practice and Experience, Journal of Grid Computing, Journal of Supercomputing, and others. For the agent research community, our approach will represent a good use case, showing how fundamental agent research can be deployed to solve problems originating in other domains. The impact will be higher, as we envisage that we will develop novel agent interaction protocols that can constitute scientific contributions for the agent research field. A better understanding of the business model of a SaaS provider will lead to increased

competition (and probably lower prices) in the service providers market, allowing other interested third parties to develop such businesses.

2.2.2.4 Methodology

We will adopt the standard computer science research methodology, starting with the identification and formalization of the research questions and environment, defining models of interest for solving various solutions, simulation of the models and extraction of the results. We will produce a prototype that will allow easy integration of different infrastructure providers, automatic establishment of pricing policies, automated negotiation of SLA parameters between the consumers and the SaaS provider and between the provider and infrastructure owners. The prototype will make transparent the deployment policy and the service management during delivery.

We base our work on the following human resources: the principal investigator, two postdoctoral researchers - denoted as DR1 and DR2, and two master graduates - denoted as MSC1 and MSC2.

We envisage the following activities:

A1. Identification and formalization of the SaaS provider environment, including the identification of efficiency properties to be studied, the cost and the computing endowments of various computing infrastructures to be considered as cloud resources

A2. Defining the formal models for optimization of the efficiency properties

A3. Defining the agent model for the population of the deployed agents, including the behaviors of CNAg(s), IOAg(s) and the PAg(s). Here, we include the definition of the interaction protocol between the PAg(s) and the IOAg(s) as a concurrent negotiation setup and the definition of the interactions between the CNAg(s) and the IOAg(s) for transparent and autonomic cloud platform management.

A4. Development of simulation experiments using various simulators for validating the models of the efficiency properties

A5. Development of an agent solution for the envisaged agent model. We will employ JADE(http://jade.tilab.com/) for agent development and deployment.

A6. Performing simulations with the models against various cloud ecosystem properties and extracting the results

A7. Implementation of a prototype, based on the conclusions extracted from the previous simulations

A8. Validation of the prototype in real scenarios and extract results from prototype validation

A9. Team management and results dissemination by writing papers to top conferences, workshops and journals - this activity will take place all the three years duration envisaged for the above-mentioned activities, for assuring timely presentation of the scientific results

Assignment of the human resources on the activities will be as following:

- DR1 will work on optimization of the efficiency properties and implementations related to the inclusion of the desktop grids and peer-to-peer services in the model
- DR2 will work on optimization of the efficiency properties and implementations related with risk management and task scheduling on private and public clouds, P2P and grid resources. He will be strongly involved in A4 and A6, as he has extended experience with cloud simulators and in prototype implementation
- MSC1 will be involved in the design and implementation of the multi-agent system, including the agent interaction protocols and the prototype implementation (A3 and A5)
- MSC2 will be involved with the implementations and deployment of the cloud infrastructures and the implementation of the prototype

The research directions presented above will be developed part of the automated collaborative systems group in the Business Information Systems department at Babeş-Bolyai University. The research group currently holds a HPC cluster with 2 IBM M3 x3620 systems, each server with 2 Intel Xeon E5620 processors, 32 Gb RAM and 1Tb Hdd. During 2012, the cluster will be extended with another 2 IBM nodes, acquired from the center resources. The cluster runs Eucalyptus, allowing for easy deployment of private clouds. The research team will use this cluster for all the experiments. The research center will offer the office for the research team members and all research infrastructure of the BBU, including the library, will be available for the members usage.

References

Abdul-Rahman, A. and Hailes, S. (2000). Supporting trust in virtual communities. In *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 6*, page 6007. IEEE Computer Society. 30, 39, 60

Aberer, K. and Despotovic, Z. (2001). Managing trust in a peer-2-peer information system. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 310–317. ACM Press. 58, 61

Adar, E. and Huberman, B. A. (2000). Free Riding on Gnutella. *First Monday*, 5(10). 10, 13, 126

Ahsant, M., Surridge, M., Leonard, T., Krishna, A., and Mulmo, O. (2006). Dynamic trust federation in grids. In *iTrust2006: Proceedings of the 4th International Conference on Trust Management*, volume 3986 of *Lecture Notes in Computer Science*, pages 3–18. Springer. 31

Anderson, D. P. (2004). BOINC: A system for public-resource computing and storage. In *5th Intl Workshop on Grid Computing (GRID 2004), 2004, USA, Proceedings*, pages 4–10. IEEE Computer Society. 21, 90, 93, 95

Anderson, D. P. and McLeod, J. (2007). Local scheduling for volunteer computing. In *21th Intl. Parallel and Distributed Processing Symposium (IPDPS 2007), Proceedings, 2007, USA*, pages 1–8. IEEE Computer Society. 90

Andrade, N., Brasileiro, F., Cirne, W., and Mowbray, M. (2007). Automatic grid assembly by promoting collaboration in peer-to-peer grids. *Journal of Parallel Distributed Computing*, 67(8):957–966. 23

Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., and Xu, M. (2007). Web Services Agreement Specification (WS-Agreement). In *OGF Grid Resource Allocation Agreement Protocol (GRAAP) WG*. http://www.ggf.org/documents/GFD.107.pdf - consulted on 13 July 2011. 3, 18, 125, 126, 131, 174

Araujo, F., Farinha, J., Domingues, P., Silaghi, G. C., and Kondo, D. (2011). A maximum independent set approach for collusion detection in voting pools. *Journal of Parallel and Distributed Computing*, 71(10):1356 – 1366. 5, 16, 89, 92, 121, 122

Arenas, A. E., Aziz, B., and Silaghi, G. C. (2010). Reputation management in collaborative computing systems. *Security and Communication Networks*, 3(6):546–564. 4, 24

Artigas, M. S., Lopez, P. G., Ahullo, J. P., and Skarmeta, A. F. G. (2005). Cyclone: A novel design schema for hierarchical dhts. In *Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing*, pages 49–56. IEEE Computer Society. 10

Austin, D., Barbir, A., Ferris, C., and Garg, S. (2004). *Web Services Architecture Requirements*. W3C Working Group. http://www.w3.org/TR/wsa-reqs/, consulted on 21 April 2012. 17

Ausubel, L. and Deneckere, R. (1993). Efficient Sequential Bargaining. *Review of Economic Studies*, 60(2):435–461. 126, 129, 141

Baarslag, T., Hindriks, K., Jonker, C. M., Kraus, S., and Lin, R. (2012). The First Automated Negotiating Agents Competition (ANAC 2010). In *New Trends in Agent-based Complex Automated Negotiations*, volume 383 of *Studies in Computational Intelligence*. Springer. 149, 150

Barry, D. K. (2003). *Web-services and service-oriented architectures: the savvy manager's guide*. Morgan Kaufmann. 17

Beloglazov, A. and Buyya, R. (2010). Energy efficient resource management in virtualized cloud data centers. In *Proceedings of the 2010 10th IEEE/ACM*

International Conference on Cluster, Cloud and Grid Computing, pages 826–831. IEEE Computer Society. 174

Birman, K. (2005). *Reliable Distributed Systems: Technologies, Web Services, and Applications*. Springer. 17

Blasi, L., Arenas, A., Aziz, B., Mori, P., Rovati, U., Crispo, B., Martinelli, F., and Massonet, P. (2008). A secure environment for grid-based supply chains. In Cunningham, P. and Cunningham, M., editors, *eChallenges 2008 Conference, Collaboration and the Knowledge Economy: Issues, Applications, Case Studies*. IOS Press. 24

Brandic, I., Music, D., and Dustdar, S. (2010). VieSLAF Framework: Facilitating Negotiations in Clouds by Applying Service Mediation and Negotiation Bootstrapping . *Scalable Computing: Practice and Experience*, 11(2):189–204. 132

Britton, D., Cass, A., Clarke, P., Coles, J., Colling, D., Doyle, A., Geddes, N., Gordon, J., Jones, R., Kelsey, D., Lloyd, S., Middleton, R., Patrick, G., Sansum, R., and Pearce, S. (2009). GridPP: the UK grid for particle physics. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1897):2447–2457. 24

Buchegger, S. and Boudec, J. L. (2005). Self-Policing Mobile Ad-Hoc Networks by Reputation. *IEEE Communication Magazine*, 43(7):101–107. 31

Buyya, R., Yeo, C. S., and Venugopal, S. (2008). Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *10th IEEE International Conference on High Performance Computing and Communications*, pages 5–13. IEEE Press. 171

Camarihna-Matos, L. M. and Afsarmanesh, H. (2003). Elements of a base ve infrastructure. *Journal of Computers in Industry*, 51(2):139–163. 68

Canon, L.-C., Jeannot, E., and Weissman, J. (2010). A dynamic approach for characterizing collusion in desktop grids. In *2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS)*, pages 1 –12. IEEE Press. 4, 89, 91, 94

Canon, L.-C., Jeannot, E., and Weissman, J. (2011). A scheduling and certification algorithm for defeating collusion in desktop grids. In *International Conference on Distributed Computing Systems*, pages 343–352. IEEE Press. 4, 91, 94

Cappello, F., Djilali, S., Fedak, G., Hérault, T., Magniette, F., Néri, V., and Lodygensky, O. (2005). Computing on large-scale distributed systems: XtremWeb architecture, programming models, security, tests and convergence with grid. *Future Generation Computer Systems*, 21(3):417–437. 21, 90, 95

Chang, I., Shao, W.-Z., and Teh, H.-H. (1988). Heuristic solutions for the general maximum independent set problem with applications to expert system design. In *Twelfth International Computer Software and Applications Conference, COMPSAC 88.*, pages 451–455. IEEE Press. 122

Cheng, W.-K., Ooi, B.-Y., and Chan, H.-Y. (2010). Resource Federation in Grid Using Automated Intelligent Agent Negotiation. *Future Generation Computer Systems*, 26(8):1116 – 1126. 127, 132

Chien, A., Calder, B., Elbert, S., and Bhatia, K. (2003). Entropia: architecture and performance of an enterprise desktop grid system. *Journal of Parallel and Distributed Computing*, 63(5):597 – 610. 21

Choi, S. and Buyya, R. (2010). Group-based adaptive result certification mechanism in desktop grids. *Future Generation Computer Systems*, 26(5):776 – 786. 93, 94

Chrabakh, W. and Wolski, R. (2006). GridSAT: a system for solving satisfiability problems using a computational grid. *Parallel Computing*, 32(9):660–687. 21

Chu, Y.-h., Chuang, J., and Zhang, H. (2004). A case for taxation in peer-to-peer streaming broadcast. In *Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, pages 205–212. ACM Press. 14

Ciccarelli, G. and Lo Cigno, R. (2011). Collusion in peer-to-peer systems. *Computer Networks*, 55(15):3517–3532. 7

Clark, K., Oey, M., Warnier, M., and Brazier, F. (2009). WS-Agreement Negotiation Perspective - draft version 2. In *OGF Grid Resource Allocation Agreement Protocol (GRAAP) WG*. http://forge.ggf.org/sf/go/doc15817?nav=1.32 - consulted on 13 July 2011. 126, 131

Cohen, B. (2003). Incentives Build Robustness in BitTorrent. http://www.bittorrent.org/bittorrentecon.pdf - consulted on 13 July 2011. 13, 126

CoreGrid (2005). D.IA.03 Survey Material on Trust and Security. Technical Report D.IA.03, CoreGrid. 26, 27, 30

Costa, F., Silva, L., Fedak, G., and Kelley, I. (2008). Optimizing data distribution in desktop grid platforms. *Parallel Processing Letters*, 18(3):391–410. 90

Das, R., Whalley, I., and Kephart, J. O. (2006). Utility-based collaboration among autonomous agents for resource allocation in data centers. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1572–1579. ACM Press. 175

Dearden, R., Friedman, N., and Russel, S. (1998). Bayesian Q-Learning. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 761–768. 134

DeFanti, T. A., Foster, I., Papka, M. E., Stevens, R., and Kuhfuss, T. (1996). Overview of the i-way: Wide-area visual supercomputing. *International Journal of High Performance Computing Applications*, 10(2-3):123–131. 19

Dellarocas, C. (2005). Reputation mechanism design in online trading environments with pure moral hazard. *Information Systems Research*, 16(2):209–230. 27

Dellarocas, C. (2006). How often should reputation mechanisms update a trader's reputation profile? *Information Systems Research*, 17(3). 37

Despotovic, Z. and Aberer, K. (2005). Probabilistic prediction of peers' performance in p2p networks. *Engineering Applications of Artificial Intelligence*, 18(3):771–780. 31, 40, 45, 61

Deutch, M. (1962). Cooperation and trust: Some theoretical notes. In *Nebraska Symposium on Motivation*, pages 275–319. Nebraska University Press. 29

Di Stefano, A., Morana, G., and Zito, D. (2009). A P2P Strategy for QoS Discovery and SLA Negotiation in Grid Environment. *Future Generation Computer Systems*, 25(8):862 – 875. 132

Domingues, P., Sousa, B., and Silva, L. M. (2007). Sabotage-tolerance and trust management in desktop grid computing. *Future Generation Computer Systems*, 23(7):904–912. 20, 90, 94, 123

Douceur, J. R. (2002). The Sybil Attack. In *Peer-to-Peer Systems, First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002, Revised Papers*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer. 15, 91

Du, W., Jia, J., Mangal, M., and Murugesan, M. (2004). Uncheatable grid computing. In *24th International Conference on Distributed Computing Systems, 2004. Proceedings.*, pages 4 – 11. IEEE Press. 93

Falcone, R. and Castelfranci, C. (2001). Social trust: A cognitive approach. In Castelfranchi, C. and Tan, Y.-H., editors, *Trust and Deception in Virtual Societibes*, pages 55–90. Kluwer Academic Publishers. 29

Faratin, P., Sierra, C., and Jennings, N. (2002). Using Similarity Criteria to Make Issue Trade-offs in Automated Negotiations. *Artificial Intelligence*, 142(2):205–237. 134, 136, 138, 139, 146

Faratin, P., Sierra, C., and Jennings, N. R. (1998). Negotiation Decision Functions for Autonomous Agents. *Robotics and Autonomous Systems*, 24:159–182. 133

Fedak, G., He, H., and Cappello, F. (2008). Bitdew: a programmable environment for large-scale data management and distribution. In *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, pages 1–12. IEEE Press. 22, 23, 90

Feldman, M., Papadimitriou, C., Chuang, J., and Stoica, I. (2004). Free-riding and whitewashing in peer-to-peer systems. In *Proceedings of the ACM SIG-*

COMM workshop on Practice and theory of incentives in networked systems, pages 228–236. ACM Press. 16

Fernandez, A., Lopez, L., Santos, A., and Georgiou, C. (2006). Reliably executing tasks in the presence of untrusted entities. In *25th IEEE Symposium on Reliable Distributed Systems, SRDS '06.*, pages 39 –50. IEEE Press. 95

Figueroa, C., Figueroa, N., Jofre, A., Sahai, A., Chen, Y., and Iyer, S. (2008). A Game Theoretic Framework for SLA Negotiation. Technical report, Enterprise Systems Storage Laboratory, HP Laboratories. http://www.hpl.hp.com/techreports/2008/HPL-2008-5.pdf, consulted on 5 August 2011. 127, 133

Fisk, A. (2003). *Gnutella Dynamic Query Protocol v0.1.* http://limewire.negatis.com/index.php?title=Dynamic_Querying. 11

Fito, J., Macias, M., and Guitart, J. (2010). Toward business-driven risk management for cloud computing. In *2010 International Conference on Network and Service Management (CNSM)*, pages 238–241. IEEE Press. 176

Fletcher, G., Sheth, H., and Brner, K. (2005). Unstructured peer-to-peer networks: Topological properties and search performance. In *Agents and Peer-to-Peer Computing*, volume 3601 of *Lecture Notes in Computer Science*, pages 14–27. Springer. 11

Foster, I. and Iamnitchi, A. (2003). On death, taxes, and the convergence of peer-to-peer and grid computing. In Kaashoek, F and Stoica, I, editor, *Peer-to-Peer Systems II, 2nd International Workshop on Peer-to-Peer Systems,*, volume 2735 of *Lecture Notes in Computer Science*, pages 118–128. Springer. 2

Foster, I., Kesselman, C., and Tuecke, S. (2001). The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222. 20, 25, 26, 65

Foster, I., Kishimoto, H., Savva, A., Berry, D., Djaoui, A., and Grimshaw, A. (2005). The open grid services architecture, version 1.0. Technical Report GFD-I.030, GGF. http://www.gridforum.org/documents/GWD-I-E/GFD-I.030.pdf, consulted on 10 January 2012. 31

Friedman, E. J. and Resnick, P. (2001). The social cost of cheap pseudonyms. *Journal of Economics and Management Strategy*, 10(2):173–199. 16

Fudenberg, D. and Levine, K. (1992). Maintaining a reputation when strategies are imperfectly observed. *Review of Economic Studies*, 59(3):561–579. 36

Fudenberg, D. and Tirole, J. (1991). *Game Theory*. MIT Press. 15, 36

Fullam, K. and Barber, K. (2006). Learning trust strategies in reputation exchange networks. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1241–1248. ACM Press. 51, 61

Fullam, K., Klos, T., Muller, G., Sabater, J., Schlosser, A., Topol, Z., Barber, K., Rosenschein, J., Vercouter, L., and Voss, M. (2004). The Agent Reputation and Trust (ART) Testbed Competition Game Rules (version 1.0). Technical Report TR2004-UT-LIPS-028, The University of Texas at Austin. 50

Fullam, K., Klos, T., Muller, G., Sabater, J., Topol, Z., Barber, K., Rosenscheim, J., and Vercouter, L. (2005). The agent reputation and trust (art) testbed architecture. In *Frontiers in Artificial Intelligence and Applications*, volume 131, pages 389–396. IOS Press. 50

Gambetta, D. (1988). *Trust: Making and Breaking Cooperative Relations*, chapter Can We Trust Trust?, pages 213–237. Department of Sociology, University of Oxford. 28, 29

Garcia-Molina, H. and Crespo, A. (2003). Semantic overlay networks for p2p systems. Technical Report 2003-75, Stanford InfoLab. http://ilpubs.stanford.edu:8090/627/. 11

Garey, M. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman. 122

Georgakopoulos, D. and Papazoglou, M. (2009). *Service-Oriented Computing*. MIT Press. 17

Gode, D. K. and Shyam, S. (1993). Allocative Efficiency of Markets with Zero-intelligence Traders: Market as a Partial Substitute for Individual Rationality. *The Journal of Political Economy*, 101(1):119–137. 134, 135

Goel, S., Talya, S. S., and Sobolewski, M. (2007). Service-based p2p overlay network for collaborative problem solving. *Decission Support Systems*, 43(2):547–568. 19

Goiri, I., Berral, J. L., Fito, J. O., Julir, F., Nou, R., Guitart, J., Gavaldr, R., and Torres, J. (2012). Energy-efficient and multifaceted resource management for profit-driven virtualized data centers. *Future Generation Computer Systems*, 28(5):718 – 731. 175

Golle, P. and Mironov, I. (2001). Uncheatable distributed computations. In Naccache, D., editor, *Topics in Cryptology CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 425–440. Springer Berlin / Heidelberg. 93

Gong, L. (2001). Jxta: A network programming environment. *IEEE Internet Computing*, 5(3):88–95. 19, 22

Gorgone, J., Gray, P., Stohr, E., Valacich, J., and Wigand, R. (2006). MSIS 2006: Model Curriculum and guidelines for graduate degree programs in Information Systems. *Communications of AIS*, 38:121–196. 159, 163, 164

Grandison, T. and Sloman, M. (2000). A Survey of Trust in Internet Applications. *IEEE Communications Surveys and Tutorials*, 3(4). 26, 27, 29

Guha, S., Rastogi, R., and Shim, K. (1999). Rock: A robust clustering algorithm for categorical attributes. In *15th International Conference on Data Engineering*, pages 512–521. IEEE Computer Society. 119

Gupta, M., Judge, P., and Ammar, M. (2003). A reputation system for peer-to-peer networks. In *NOSSDAV '03: Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, pages 144–152. ACM Press. 31, 53, 54, 55, 61, 67

Habib, A. and Chuang, J. (2004). Incentive mechanism for peer-to-peer media streaming. In *12th IEEE International Workshop on Quality of Service*, pages 171 – 180. IEEE Press. 14

Habib, S., Ries, S., and Muhlhauser, M. (2011). Towards a trust management system for cloud computing. In *2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 933–939. IEEE Press. 176

Haque, A., Alhashmi, S. M., and Parthiban, R. (2011). A Survey of Economic Models in Grid Computing. *Future Generation Computer Systems*, 27(8):1056 – 1069. 133

Hindriks, K., Jonker, C. M., Kraus, S., Lin, R., and Tykhonov, D. (2009). Genius: Negotiation Environment for Heterogeneous Agents. In *AAMAS '09: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 1397–1398. IFAAMAS. 134, 146

Hindriks, K. and Tykhonov, D. (2008). Opponent Modelling in Automated Multi-issue Negotiation Using Bayesian Learning. In *AAMAS '08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*. 125, 127, 134, 135, 136, 137, 138, 146

Huynh, T., Jennings, N. R., and Shadbolt, N. (2006). An integrated trust and reputation model for open multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 13(2):119–154. 44, 45, 49, 60, 67, 73

IFIP Working Group 10.4 (1988). *Charter of WG 10.4 Dependable Computing and Fault Tolerance*. 3

Irwin, D., Grit, L., and Chase, J. (2004). Balancing risk and reward in a market-based task service. In *13th IEEE International Symposium on High performance Distributed Computing*, pages 160 – 169. IEEE Press. 176

Ito, T., Zhang, M., Robu, V., Fatima, S., and Matsuo, T. (2012). *New Trends in Agent-based Complex Automated Negotiations*. Studies in Computational Intelligence. Springer. 151

Jain, M., Lal, S., and Mathuria, A. (2008). A survey of peer-to-peer micropayment schemes. Technical report, DA-IICT. http://www.dgp.toronto.edu/m̃jain/P2P_Micropayment-2008.pdf consulted on 19 April 2012. 14

Jian, L. (2008). An Agent Bilateral Multi-issue Alternate Bidding Negotiation Protocol Based on Reinforcement Learning and its Application in E-commerce. In *ISECS '08: Proceedings of the 2008 International Symposium on Electronic Commerce and Security*, pages 217–220. IEEE Computer Society. 134

Jones, S. and Morris, P. (1999). TRUST-EC: requirements for Trust and Confidence in E-Commerce. Technical Report EUR 18749 EN, European Commission Joint Research Centre. 29

Jonker, C. M., Robu, V., and Treur, J. (2007). An Agent Architecture for Multi-attribute Negotiation Using Incomplete Preference Information. *Autonomous Agents and Multi-Agent Systems*, 15(2):221–252. 134, 135, 138, 146

Jonker, C. M. and Treur, J. (2001). An Agent Architecture for Multi-Attribute Negotiation. In *IJCAI'01: Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 1195–1201. Morgan Kaufmann. 134

Jøsang, A. (1999). An algebra for assessing trust in certification chains. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 1999, San Diego, California, USA*. The Internet Society. 43, 60, 67

Jøsang, A., Ismail, R., and Boyd, C. (2007). A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618 – 644. 26, 27, 29, 30, 38

Jøsang, A. and Knapskog, S. (1998). A metric for trusted systems. In *Proceedings of the 21st National Information Systems Security Conference, (NIST-NCSC 1998)*. 42

Jurca, R. and Faltings, B. (2003). An incentive compatible reputation mechanism. In *2003 IEEE International Conference on E-Commerce Technology*, page 285. IEEE Computer Society. 61, 67

Jurca, R. and Faltings, B. (2005a). Reputation-based pricing of p2p services. In *P2PECON '05: Proceeding of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, pages 144–149. ACM Press. 31, 63

Jurca, R. and Faltings, B. (2005b). Reputation-based service level agreements for web services. In Benatallah, B., Casati, F., and Traverso, P., editors, *Service-Oriented Computing - ICSOC 2005*, volume 3826 of *Lecture Notes in Computer Science*, pages 396–409. Springer. 31, 63, 67

Kaashoek, M. F. and Karger, D. R. (2003). Koorde: A Simple Degree-Optimal Distributed Hash Table. In *Proceedings of the IPTPS 2003*, pages 98–107. 10

Kafali, O. and Yolum, P. (2006). Trust strategies for art testbed. In *The workshop Trust in Agent Societies at AAMAS 2006*. 51, 61

Kamvar, S., Schlosser, M., and Garcia-Molina, H. (2003). The eigentrust algorithm for reputation management in p2p networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 640–651. ACM Press. 15, 31, 55, 56, 60, 64, 65, 66, 106, 107, 118

Karlin, S. and Taylor, H. M. (1975). *A First Course in Stochastic Processes, Second Edition*. Academic Press. 103

Kelley, I. and Taylor, I. (2008). Bridging the data management gap between service and desktop grids. In *Distributed and Parallel Systems*, pages 13–26. Springer US. 23

Kerschbaum, F., Haller, J., Karabulut, Y., and Robinson, P. (2006). Pathtrust: A trust-based reputation service for virtual organization formation. In *iTrust2006: Proceedings of the 4th International Conference on Trust Management*, volume 3986 of *Lecture Notes in Computer Science*, pages 193–205. Springer. 31, 60, 65

Kim, J.-S., Nam, B., Marsh, M. A., Keleher, P. J., Bhattacharjee, B., Richardson, D., Wellnitz, D., and Sussman, A. (2007). Creating a robust desktop grid using peer-to-peer services. In *21th Intl. Parallel and Distributed Processing Symposium (IPDPS 2007), Proceedings, 2007, USA*. IEEE. 90

Kliazovich, D., Bouvry, P., and Khan, S. (2010). Greencloud: a packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing*, page onlie first. 175

Kondo, D., Araujo, F., Malecot, P., Domingues, P., Silva, L. M., Fedak, G., and Cappello, F. (2007). Characterizing result errors in internet desktop grids. In *Euro-Par 2007, Parallel Processing, 13th International Euro-Par Conference, France, 2007, Proceedings*, volume 4641 of *LNCS*, pages 361–371. Springer. 90, 92, 93, 97, 98, 100

Kreps, D. and Wilson, R. (1982). Reputation and imperfect information. *Journal of Economic Theory*, 27(2):253–279. 36

Kreps, D. M. (1990). *A course in microeconomic theory*. Princeton University Press. 14

Krishnan, R., Smith, M. D., and Telang, R. (2006). The economics of peer-to-peer networks. *Journal of Information Technology Theory and Application*, 5(3). 19

Lai, K., Rasmusson, L., Adar, E., Zhang, L., and Huberman, B. A. (2005). Tycoon: An implementation of a distributed, market-based resource allocation system. *Multiagent and Grid Systems*, 1(3):169–182. 14

Lamanna, D. D., Skene, J., and Emmerich, W. (2003). Slang: A language for defining service level agreements. In *Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems*, pages 100–106. IEEE Computer Society. 19

Lamport, L., Shostak, R., and Pease, M. (1982). The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4:382–401. 95

Lang, F. (2005). Developing Dynamic Strategies for Multi-Issue Automated Contracting in the Agent Based Commercial Grid. In *IEEE International Symposium on Cluster Computing and the Grid*, pages 342–349. IEEE Computer Society. 127, 133, 141

Lawley, R., Luck, M., Decker, K., Payne, T. R., and Moreau, L. (2003). Automated Negotiation Between Publishers and Consumers of Grid Notifications. *Parallel Processing Letters*, 13(4):537–548. 127

Lee, Y. and Zomaya, A. (2012). Energy efficient utilization of resources in cloud computing systems. *The Journal of Supercomputing*, 60(2):268–280. 175

Legrand, A., Marchal, L., and Casanova, H. (2003). Scheduling Distributed Applications: the SimGrid Simulation Framework. In *CCGRID '03: Proc. of the 3st International Symposium on Cluster Computing and the Grid*, pages 138–145. IEEE Computer Society. 82

Li, C. and Li, L. (2004). Competitive Proportional Resource Allocation Policy for Computational Grid. *Future Generation Computer Systems*, 20(6):1041 – 1054. 127

Li, J., Chinneck, J., Woodside, M., and Litoiu, M. (2009). Fast scalable optimization to configure service systems having cost and quality of service constraints. In *Proceedings of the 6th international conference on Autonomic computing*, ICAC '09, pages 159–168. ACM Press. 175

Li, J., Sim, K. M., and Yahyapour, R. (2007). Negotiation Strategies Considering Opportunity Functions for Grid Scheduling. In Kermarrec, AM and Bouge, L and Priol, T, editor, *Euro-Par 2007 Parallel Processing, Proceedings*, volume 4641 of *Lecture Notes in Computer Science*, pages 447–456. Springer. 127, 133

Li, J. and Yahyapour, R. (2006). A Strategic Negotiation Model for Grid Scheduling. *International Transactions on Systems Science and Applications*, 1(4):411–420. 127, 132, 141

Li, Y.-M. and Lee, Y.-L. (2010). Pricing peer-produced services: Quality, capacity, and competition issues. *European Journal of Operational Research*, 207(3):1658 – 1668. 20

Lim, S.-H., Sharma, B., Nam, G., Kim, E. K., and Das, C. (2009). MDCSim: A multi-tier data center simulation, platform. In *IEEE International Conference on Cluster Computing and Workshops, 2009. CLUSTER '09*, pages 1–9. IEEE Press. 176

Litoiu, M. and Litoiu, M. (2010). Optimizing resources in cloud, a soa governance view. In *Proceedings of the 2010 Workshop on Governance of Technology, Information and Policies*, pages 71–75. ACM. 174

Lopes, F., Mamede, N., Novais, A., and Coelho, H. (2004). Negotiation Strategies for Autonomous Computational Agents. In *ECAI 2004: Proceedings of the 16th European Conference on Artificial Intelligence*, pages 38–42. IOS Press. 134

Ludwig, H., Keller, A., Dan, A., King, R. P., and Franck, R. (2003). *Web Service Level Agreement (WSLA) Language Specification*. IBM Corporation. http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf consulted on 23 April 2012. 19

Lv, Q., Cao, P., Cohen, E., Li, K., and Shenker, S. (2002). Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th international conference on Supercomputing*, pages 84–95. ACM Press. 11

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In Cam, L. M. L. and Neyman, J., editors, *Proc. of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press. 91, 119

Manchala, D. (2000). E-commerce trust metrics and models. *IEEE Internet Computing*, 4(2):36–44. 27

Marozzo, F., Talia, D., and Trunfio, P. (2011). P2p-mapreduce: Parallel data processing in dynamic cloud environments. *Journal of Computer and System Sciences*. 22

Marsh, S. (1994). *Formalizing Trust as a Computational Concept*. PhD thesis, University of Stirling. 29, 42

Marzolla, M., Mordacchini, M., and Orlando, S. (2007). Peer-to-peer systems for discovering resources in a dynamic grid. *Parallel Computing*, 33(4-5):339–358. 12

Mastroianni, C., Talia, D., and Verta, O. (2005). A super-peer model for resource discovery services in large-scale Grids. *Future Generation Computer Systems*, 21(8):1235 – 1248. 12

Maurer, M., Brandic, I., Emeakaroha, V. C., and Dustdar, S. (2010). Towards knowledge management in self-adaptable clouds. In *Proceedings of the 2010 6th*

World Congress on Services, SERVICES '10, pages 527–534. IEEE Computer Society. 175

Maymounkov, P. and Mazieres, D. (2002). Kademlia: A peer-to-peer information system based on the xor metric. In *Peer-to-Peer Systems*, volume 2429 of *Lecture Notes in Computer Science*, pages 53–65. Springer Berlin / Heidelberg. 9

Meshkova, E., Riihijärvi, J., Petrova, M., and Mähönen, P. (2008). A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks. *Computer Networks*, 52(11):2097–2128. 7

Miller, C., Butler, P., Shah, A., and Butt, A. R. (2007). Peerstripe: a p2p-based large-file storage for desktop grids. In *Proceedings of the 16th international symposium on High performance distributed computing*, pages 221–222. ACM Press. 23

Milojicic, D. S., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S., and Xu, Z. (2003). Peer-to-peer computing. Technical Report HPL-2002-5, HP Laboratories Palo Alto. 6

Moca, M. (2010). *Issues towards Automated Collaboration in Distributed Computing in Business*. Risoprint. viii, 9

Moca, M., Silaghi, G. C., and Fedak, G. (2011). Distributed Results Checking for MapReduce in Volunteer Computing. In *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW)*, pages 1847–1854. IEEE Press. 95

Montresor, A. and Abeni, L. (2011). Cloudy weather for p2p, with a chance of gossip. In *2011 IEEE International Conference on Peer-to-Peer Computing (P2P)*, pages 250–259. IEEE Press. 177

Moon, H. J., Chi, Y., and Hacigümüs, H. (2010). Sla-aware profit optimization in cloud services via resource scheduling. In *Proceedings of the 2010 6th World Congress on Services*, pages 152–153. IEEE Computer Society. 175

Mowbray, M. (2007). How web community organisation can help grid computing. *International Journal of Web Based Communities*, 3(1):44–54. 21, 23

Newmarch, J. (2006). *Foundations of Jini 2 Programming*. Apress. 19

Nguyen, T. D. and Jennings, N. R. (2003). A Heuristic Model for Concurrent Bi-lateral Negotiations in Incomplete Information Settings. In *IJCAI'03: Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 1467–1469. Morgan Kaufmann. 133

Palomar, E., Estevez-Tapiador, J., Hernandez-Castro, J., and Ribagorda, A. (2006). Security in p2p networks: Survey and research directions. In *Emerging Directions in Embedded and Ubiquitous Computing*, volume 4097 of *Lecture Notes in Computer Science*, pages 183–192. Springer Berlin / Heidelberg. 7

Paschke, A. (2005). Rbsla a declarative rule-based service level agreement language based on ruleml. In *International Conference on Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, volume 2, pages 308–314. IEEE Press. 19

Paton, N. W., Aragão, M. A. T., Lee, K., Fernandes, A. A. A., and Sakellariou, R. (2009). Optimizing utility in cloud computing through autonomic workload execution. *IEEE Data Engineering Bulletin*, 32(1):51–58. 175

Petri, I., Rana, O. F., and Silaghi, G. C. (2011). Service level agreement as a complementary currency in peer-to-peer markets. *Future Generation Computer Systems*. 14

Pichot, A., Waeldrich, O., Ziegler, W., and Wieder, P. (2009). Towards Dynamic Service Level Agreement Negotiation: an Approach Based on WS-Agreement. In *Web Information Systems and Technologies*, volume 18 of *Lecture Notes in Business Information Processing*, pages 107–119. Springer. 125, 127

Popovici, F. I. and Wilkes, J. (2005). Profitable services in an uncertain world. In *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, SC '05. IEEE Computer Society. 176

Puppin, D., Moncelli, S., Baraglia, R., Tonellotto, N., and Silvestri, F. (2005). A grid information service based on peer-to-peer. In *Euro-Par 2005 Parallel*

Processing, volume 3648 of *Lecture Notes in Computer Science*, pages 613–613. Springer Berlin / Heidelberg. 12

Raeesy, Z., Brzostowski, J., and Kowalczyk, R. (2007). Towards a Fuzzy-Based Model for Human-like Multi-Agent Negotiation. In *Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 515–519. IEEE Computer Society. 134

Ramchurn, S., Huynh, D., and Jennings, N. (2004a). Trust in multi-agent systems. *Knowledge Engineering Review*, 19(1):1–25. 28, 30

Ramchurn, S., Jennings, N., Sierra, C., and Godo, L. (2004b). Devising a trust model for multi-agent interactions using confidence and reputation. *Applied Artificial Intelligence*, 18(9-10):833–852. 47, 50, 52, 61, 67, 73

Ranjan, R. and Zhao, L. (2011). Peer-to-peer service provisioning in cloud computing environments. *The Journal of Supercomputing*, page online first. 175

Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S. (2001). A scalable content-addressable network. *SIGCOMM Computer Communication Review*, 31:161–172. 8

Rebahi, Y., Mujica, V., and Sisalem, D. (2005). A reputation-based trust mechanism for ad hoc networks. In *ISCC '05: Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC'05)*, pages 37–42. IEEE Computer Society. 31

Resende, M. G. C., Feo, T. A., and Smith, S. H. (1998). Algorithm 787: Fortran subroutines for approximate solution of maximum independent set problems using grasp. *ACM Transactions on Mathematical Software*, 24:386–394. 122

Resnick, P., Kuwabara, K., Zeckhauser, R., and Friedman, E. (2000). Reputation systems. *Communications of ACM*, 43(12):45–48. 27

Resnick, P. and Zeckhauser, R. (2002). *Advances in Applied Microeconomics*, volume 11, chapter Trust Among Strangers in Internet Transactions: Empirical Analysis of eBay's Reputation System. Elsevier. 37, 48

Resnick, P., Zeckhauser, R., Swanson, J., and Lockwood, K. (2006). The value of reputation on ebay: A controlled experiment. *Experimental Economics*, 9(2):79–101. 38

Risson, J. and Moors, T. (2006). Survey of research towards robust peer-to-peer networks: search methods. *Computer Networks*, 50(17):3485–3521. 7

Rochwerger, B., Breitgand, D., Levy, E., Galis, A., Nagin, K., Llorente, I. M., Montero, R., Wolfsthal, Y., Elmroth, E., Cáceres, J., Ben-Yehuda, M., Emmerich, W., and Galán, F. (2009). The RESERVOIR model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, 53(4):535–545. 174

Rodero-Merino, L., Anta, A. F., López, L., and Cholvi, V. (2009). Self-managed topologies in p2p networks. *Computer Networks*, 53(10):1722–1736. 11

Rosen, M., Lublinsky, B., Smith, K. T., and Balcer, M. J. (2008). *Applied SOA: Service-Oriented Architecture and Design Strategies*. Wiley. 18

Rowstron, A. I. T. and Druschel, P. (2001). Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, pages 329–350. Springer-Verlag. 9

Rubach, P. and Sobolewski, M. (2009). Dynamic SLA Negotiation in Autonomic Federated Environments. In Meersman, R., Herrero, P., and Dillon, T., editors, *On the Move to Meaningful Internet Systems: OTM 2009 Workshops*, volume 5872 of *Lecture Notes in Computer Science*, pages 248–258. Springer Berlin / Heidelberg. 132

Sabater, J. and Sierra, C. (2001). REGRET: A Reputation Model for Gregarious Societies. In *Fourth Workshop on Deception, Fraud and Trust in Agent Societies*, pages 194–195. ACM Press. 45, 47, 48, 50, 52, 61, 67, 73

Sabater, J. and Sierra, C. (2005). Review on computational trust and reputation models. *Artificial Intelligence Review*, 24(1):33–60. 28, 30

Saito, K. (2006). *i-WAT: The Internet WAT System – An Architecture for Maintaining Trust and Facilitating Peer-to-Peer Barter Relationships.* PhD thesis, Keio University. 14

Salehi, M. A. and Buyya, R. (2010). Adapting market-oriented scheduling policies for cloud computing. In *Proceedings of the 10th international conference on Algorithms and Architectures for Parallel Processing - Volume Part I*, volume 6081 of *Lecture Notes in Computer Science*, pages 351–362, Berlin, Heidelberg. Springer-Verlag. 174

Sarmenta, L. F. and Hirano, S. (1999). Bayanihan: building and studying web-based volunteer computing systems using java. *Future Generation Computer Systems*, 15(56):675 – 686. 21

Sarmenta, L. F. G. (2002). Sabotage-tolerance mechanisms for volunteer computing systems. *Future Generation Computer Systems*, 18(4):561–572. 91, 93, 96, 117

Schulz, S., Blochinger, W., Held, M., and Dangelmayr, C. (2008). COHESION A microkernel based Desktop Grid platform for irregular task-parallel applications. *Future Generation Computer Systems*, 24(5):354 – 370. 22

Sedaghat, M., Hernández, F., and Elmroth, E. (2011). Unifying cloud management: Towards overall governance of business level objectives. In *Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 591–597. IEEE Computer Society. 174

Sen, S., Goswami, I., and Airiau, S. (2006). Expertise and trust-based formation of effective coalitions: an evaluation of art testbed. In *The workshop Trust in Agent Societies at AAMAS 2006.* 51, 52

Sharma, R. K., Bash, C. E., Patel, C. D., Friedrich, R. J., and Chase, J. S. (2005). Balance of power: Dynamic thermal management for internet data centers. *IEEE Internet Computing*, 9(1):42–49. 174

Sherwood, R., Seungjoon, L., and Bhattacharjee, B. (2006). Cooperative peer groups in nice. *Computer Networks*, 50(4):523–544. 45, 59, 65, 66, 67

Siddiqui, M., Villazón, A., and Fahringer, T. (2006). Grid Capacity Planning with Negotiation-based Advance Reservation for Optimized QoS. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*. ACM. 127, 133

Silaghi, G. C., Araujo, F., Silva, L., Domingues, P., and Arenas, A. (2008a). Defeating colluding nodes in desktop grid computing platforms. In *IEEE International Symposium on Parallel and Distributed Processing, 2008. IPDPS 2008.*, pages 1 –8. IEEE Press. 4, 89, 91

Silaghi, G. C., Araujo, F., Silva, L., Domingues, P., and Arenas, A. (2009). Defeating colluding nodes in desktop grid computing platforms. *Journal of Grid Computing*, 7:555–573. 4, 89, 91, 94, 123, 176

Silaghi, G. C., Arenas, A., and Silva, L. (2007a). Reputation-based trust management systems and their applicability to grids. Technical Report TR-0064, Institutes on Knowledge and Data Management and System Architecture, CoreGRID - Network of Excellence. 4

Silaghi, G. C., Arenas, A. E., and Silva, L. M. (2007b). A utility-based reputation model for service-oriented computing. In Priol, T. and Vanneschi, M., editors, *Towards Next Generation Grids*, pages 63–72. Springer US. 4, 18, 24, 25, 69

Silaghi, G. C., Serban, L. D., and Litan, C. M. (2010). A Framework for Building Intelligent SLA Negotiation Strategies under Time Constraints. In Altmann, J. and Rana, O., editors, *GECON 2010*, volume 6296 of *Lecture Notes in Computer Science*, pages 48–61. 5, 125

Silaghi, G. C., Serban, L. D., and Litan, C. M. (2011). A Time-Constrained SLA Negotiation Strategy in Competitive Computational Grids. *Future Generation Computer Systems*. online first. 5, 16, 19, 125

Silaghi, G. C., Silva, L., Domingues, P., and Arenas, A. E. (2008b). Tackling the collusion threat in p2p-enhanced internet desktop grids. In Danelutto, M., Fragopoulou, P., and Getov, V., editors, *Making Grids Work*, pages 393–402. Springer US. 89

Silva, J. a. N., Veiga, L., and Ferreira, P. (2008). Heuristic for resources allocation on utility computing infrastructures. In *Proceedings of the 6th international workshop on Middleware for grid computing*, pages 9:1–9:6. ACM Press. 175

Sim, K. M. (2002). A MarketDriven Model for Designing Negotiation Agents. *Computational Intelligence*, 18(4):618–637. 134, 139

Sim, K. M. (2010). Grid Resource Negotiation: Survey and New Directions. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 40(3):245–257. 125, 133

Singh, A. and Liu, L. (2003). Trustme: Anonymous management of trust relationships in decentralized p2p systems. In *P2P '03: Proceedings of the 3rd International Conference on Peer-to-Peer Computing*, pages 142–149. IEEE Computer Society. 31, 53, 54, 61

Singh, M., Yu, B., and Venkatraman, M. (2001). Community-based service location. *Communications of the ACM*, 44(4):49–54. 38, 39, 44, 61, 65

SLA@SOI Consortium (2008). SLA@SOI - Overview. document available at http://sla-at-soi.eu/publications/ - consulted on 13 July 2011. 125, 174

Sobolewski, M. (2002). Fiper: The federated s2s environment. In *JavaOne, Suns 2002 Worldwide Java Developer Conference*. http://sorcersoft.org/publications/papers/2420.pdf consulted on 23 April 2012. 19

Staab, E. and Engel, T. (2009). Collusion detection for grid computing. In *9th IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009. CCGRID '09.*, pages 412 –419. IEEE Press. 4, 91, 94

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM Computer Communication Review*, 31:149–160. 8

Suryanarayana, G., Taylor, R. N., Suryanarayana, G., Taylor, R. N., Suryanarayana, G., and Taylor, R. N. (2004). A survey of trust

management and resource discovery technologies in peer-to-peer applications. Technical Report UCI-ISR-04-6, Institute of Software Research, UC Irvine. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.59.828, consulted on 10 January 2012. 27, 32, 46, 47

Talia, D. and Trunfio, P. (2005). Peer-to-peer protocols and grid services for resource discovery on grids. In *Grid Computing The New Frontier of High Performance Computing*, volume 14 of *Advances in Parallel Computing*, pages 83 – 103. North-Holland. 12

Tanenbaum, A. S. and van Steen, M. (2001). *Distributed Systems: Principles and Paradigms*. Prentice Hall. 6

Tang, B., Moca, M., Chevalier, S., He, H., and Fedak, G. (2010). Towards mapreduce for desktop grid computing. In *2010 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, pages 193 – 200. IEEE Press. 22, 90, 95

Taufer, M., Anderson, D., Cicotti, P., and Brooks, C. (2005). Homogeneous redundancy: a technique to ensure integrity of molecular simulation results using public computing. In *19th IEEE International Parallel and Distributed Processing Symposium*, page 119a. 93

Thain, D., Tannenbaum, T., and Livny, M. (2005). Distributed computing in practice: the condor experience. *Concurrency and Computation - Practice and Experience*, 17(2-4):323–356. 21

The Joint Task Force for Computing Curricula 2005 (2005). *Computing curricula 2005; The overview report coveing undergraduate degree programs in Computer Engineering, Computer Science, Information Systems, Information Technology and Software Engineering*. The Association for Computing Machinery (ACM), The Association for Information Systems (AIS), The Computer Society (IEEE-CS). 158, 161

Topi, H., Valacich, J., Wright, R., Kaiser, K., Nunamaker, J., Sipior, J. C., and de Vreede, G. (2010). *IS 2010: Curriculum Guidelines for Undergraduate Degree Programs in Information Systems* . Association for Computing Machinery (ACM) and Association for Information Systems (AIS). 166

Trunfio, P., Talia, D., Papadakis, H., Fragopoulou, P., Mordacchini, M., Pennanen, M., Popov, K., Vlassov, V., and Haridi, S. (2007). Peer-to-Peer resource discovery in Grids: Models and systems. *Future Generation Computer Systems*, 23(7):864–878. 6, 7

Tsvetovat, M. and Sycara, K. (2000). Customer coalitions in the electronic marketplace. In *Proceedings of the 4th International Conference on Autonomous Agents*, pages 263–264. ACM. 95

Vishnumurthy, V., Chandrakumar, S., and Sirer, E. (2003). KARMA: A Secure Economic Framework for Peer-to-Peer Resource Sharing. In *Workshop on the Economics of Peer-to-Peer Systems*. 14

von Laszewski, G., Alunkal, B., and Veljkovic, I. (2005). Towards reputable grids. *Scalable Computing: Practice and Experience*, 6(3):95–106. 31, 61, 64, 65

von Neumann, J. and Morgenstern, O. (1944). *Theory of Games and Economic Behavior*. Princeton University Press. 129

Vragov, R. (2005). Implicit Consumer Collusion in Auctions on the Internet. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences, 2005. HICSS '05.*, page 174c. IEEE Press. 95

Waeldrich, O., Battre, D., Brazier, F., Clark, K., Oey, M., Papaspyrou, A., Wieder, P., and Ziegler, W. (2011). WS-Agreement Negotiation 1.0. In *OGF Grid Resource Allocation Agreement Protocol (GRAAP) WG*. http://www.gridforum.org/Public_Comment_Docs/Documents/2011-03/WS-Agreement-Negotiation+v1.0.pdf - consulted on 13 July 2011. 4, 18, 126, 131

Wang, Y. and Wei, J. (2011). VIAF: Verification-Based Integrity Assurance Framework for MapReduce. In *2011 IEEE International Conference on Cloud Computing (CLOUD)*, pages 300–307. IEEE Press. 95

Watanabe, K., Fukushi, M., and Horiguchi, S. (2009a). Collusion-resistant sabotage-tolerance mechanisms for volunteer computing systems. In *IEEE International Conference on e-Business Engineering, ICEBE 2009*, pages 213–218. IEEE Press. 91

Watanabe, K., Fukushi, M., and Horiguchi, S. (2009b). Optimal spot-checking for computation time minimization in volunteer computing. *Journal of Grid Computing*, 7:575–600. 93

Watkins, C. J. C. H. and Dayan, P. (1992). Q-Learning. *Machine Learning*, 8(3-4):279–292. 132

Wei, B., Fedak, G., and Cappello, F. (2005). Collaborative data distribution with bittorrent for computational desktop grids. In *4th Intl Symposium on Parallel and Distributed Computing (ISPDC 2005), 2005, France*, pages 250–257. IEEE Computer Society. 90

Wei, W., Du, J., Yu, T., and Gu, X. (2009). SecureMR: A Service Integrity Assurance Framework for MapReduce. In *Annual Computer Security Applications Conference, 2009. ACSAC '09.*, pages 73–82. IEEE Press. 95

WepiwÉ, G. and Simeonov, P. L. (2006). Hipeer: A highly reliable p2p system. *IEICE - Transactions on Information and Systems*, E89-D:570–580. 10

White, T. (2009). *Hadoop: The Definitive Guide*. O'Reilly. 22

Wishart, R., Robinson, R., Indulska, J., and Jøsang, A. (2005). Superstringrep: reputation-enhanced service discovery. In *ACSC '05: Proceedings of the Twenty-eighth Australasian conference on Computer Science*, pages 49–57. Australian Computer Society, Inc. 43, 61

Witten, I. E. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*. Morgan Kaufmann. 110

Wong, S. (2006). An authentication protocol in web-computing. In *20th International Parallel and Distributed Processing Symposium, IPDPS 2006.*, page 10 pp. IEEE Press. 94

Woodcock, J. and Davies, J. (1996). *Using Z: Specification, Refinement, and Proof*. Prentice-Hall, Inc. 69

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems - Second Edition*. John Wiley and Sons. 2

Wu, L., Garg, S. K., and Buyya, R. (2011). Sla-based resource allocation for software as a service provider (saas) in cloud computing environments. In *Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 195–204. IEEE Computer Society. 175

Xiong, L. and Liu, L. (2004). Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Transactions on Knowledge and Data Engineering*, 16(7):843–857. 31, 56, 61, 67

Yan, J., Kowalczyk, R., Lin, J., Chhetri, M. B., Goh, S. K., and Zhang, J. (2007). Autonomous Service Level Agreement Negotiation for Service Composition Provision. *Future Generation Computer Systems*, 23(6):748 – 759. 125, 127, 133, 141

Yeo, C. S. and Buyya, R. (2009). Integrated risk analysis for a commercial computing service in utility computing. *Journal of Grid Computing*, 7(1):1–24. 176

Yokoo, M., Conitzer, V., Sandholm, T., Ohta, N., and Iwasaki, A. (2005). Coalitional games in open anonymous environments. In *Proceedings of the 20th National Conference on Artificial intelligence - Volume 2*, pages 509–514. AAAI Press. 95

Yokoo, M., Sakurai, Y., and Matsubara, S. (2004). The effect of false-name bids in combinatorial auctions: new fraud in Internet auctions. *Games and Economic Behavior*, 46(1):174–188. 95

Yu, B. and Singh, M. (2002). An evidential model of distributed reputation management. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 294–301. ACM Press. 30, 43, 49, 55, 61

Zacharia, G. and Maes, P. (2000). Trust management through reputation mechanisms. *Applied Artificial Intelligence*, 14(9):881–907. 27, 45

Zhao, B., Huang, L., Stribling, J., Rhea, S., Joseph, A., and Kubiatowicz, J. (2004). Tapestry: a resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41 – 53. 9

Zhao, H., Liu, X., and Li, X. (2011). A taxonomy of peer-to-peer desktop grid paradigms. *Cluster Computing*, 14:129–144. 22

Zhao, S., Lo, V., and GauthierDickey, C. (2005). Result verification and trust-based scheduling in peer-to-peer grids. In *P2P '05: Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing (P2P'05)*, pages 31–38. IEEE Computer Society. 31, 67, 90, 91, 93

Zhou, D. and Lo, V. (2004). Cluster computing on the fly: resource discovery in a cycle sharing peer-to-peer system. In *IEEE International Symposium on Cluster Computing and the Grid*, pages 66 – 73. IEEE Press. 22

Zulkernine, F., Martin, P., Craddock, C., and Wilson, K. (2009). A Policy-Based Middleware for Web Services SLA Negotiation. In *Proceedings of the 2009 IEEE International Conference on Web Services*, pages 1043–1050. IEEE Computer Society. 127, 133, 141