Studies in Computational Intelligence, Volume 298

Further volumes of this series can be found on our homepage: springer.com

Vol. 276. Jacek Mańdziuk
Knowledge-Free and Learning-Based Methods in Intelligent Game Playing, 2010
ISBN 978-3-642-11677-3

Vol. 277. Filippo Spagnolo and Benedetto Di Paola (Eds.)
European and Chinese Cognitive Styles and their Impact on Teaching Mathematics, 2010
ISBN 978-3-642-11679-7

Vol. 278. Radomir S. Stankovic and Jaakko Astola
From Boolean Logic to Switching Circuits and Automata, 2010
ISBN 978-3-642-11681-0

Vol. 279. Manolis Wallace, Ioannis E. Anagnostopoulos, Phivos Mylonas, and Maria Bielikova (Eds.)
Semantics in Adaptive and Personalized Services, 2010
ISBN 978-3-642-11683-4

Vol. 280. Chang Wen Chen, Zhu Li, and Shiguo Lian (Eds.)
Intelligent Multimedia Communication: Techniques and Applications, 2010
ISBN 978-3-642-11685-8

Vol. 281. Robert Babuska and Frans C.A. Groen (Eds.)
Interactive Collaborative Information Systems, 2010
ISBN 978-3-642-11687-2

Vol. 282. Husrev Taha Sencar, Sergio Velastin, Nikolaos Nikolaidis, and Shiguo Lian (Eds.)
Intelligent Multimedia Analysis for Security Applications, 2010
ISBN 978-3-642-11754-1

Vol. 283. Ngoc Thanh Nguyen, Radoslaw Katarzyniak, and Shyi-Ming Chen (Eds.)
Advances in Intelligent Information and Database Systems, 2010
ISBN 978-3-642-12089-3

Vol. 284. Juan R. González, David Alejandro Pelta, Carlos Cruz, Germán Terrazas, and Natalio Krasnogor (Eds.)
Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), 2010
ISBN 978-3-642-12537-9

Vol. 285. Roberto Cipolla, Sebastiano Battiato, and Giovanni Maria Farinella (Eds.)
Computer Vision, 2010
ISBN 978-3-642-12847-9

Vol. 286. Zeev Volkovich, Alexander Bolshoy, Valery Kirzhner, and Zeev Barzily
Genome Clustering, 2010
ISBN 978-3-642-12951-3

Vol. 287. Dan Schonfeld, Caifeng Shan, Dacheng Tao, and Liang Wang (Eds.)
Video Search and Mining, 2010
ISBN 978-3-642-12899-8

Vol. 288. I-Hsien Ting, Hui-Ju Wu, Tien-Hwa Ho (Eds.)
Mining and Analyzing Social Networks, 2010
ISBN "Pending"

Vol. 289. Anne Håkansson, Ronald Hartung, and Ngoc Thanh Nguyen (Eds.)
Agent and Multi-agent Technology for Internet and Enterprise Systems, 2010
ISBN "Pending"

Vol. 290. Weiliang Xu and John Bronlund
Mastication Robots, 2010
ISBN "Pending"

Vol. 291. Shimon Whiteson
Adaptive Representations for Reinforcement Learning, 2010
ISBN "Pending"

Vol. 292. Fabrice Guillet, Gilbert Ritschard, Henri Briand, Djamel A. Zighed (Eds.)
Advances in Knowledge Discovery and Management, 2010
ISBN "Pending"

Vol. 293. Anthony Brabazon, Michael O'Neill, and Dietmar Maringer (Eds.)
Natural Computing in Computational Finance, 2010
ISBN "Pending"

Vol. 294. Manuel F.M. Barros, Jorge M.C. Guilherme, and Nuno C.G. Horta
Analog Circuits and Systems Optimization based on Evolutionary Computation Techniques, 2010
ISBN 978-3-642-12345-0

Vol. 295. Roger Lee (Ed.)
Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2010
ISBN 978-3-642-13264-3

Vol. 296. Roger Lee (Ed.)
Software Engineering Research, Management and Applications, 2010
ISBN 978-3-642-13272-8

Vol. 297. XXX

Vol. 298. Adam Wierzbicki
Trust and Fairness in Open, Distributed Systems, 2010
ISBN 978-3-642-13450-0

Adam Wierzbicki

Trust and Fairness in Open, Distributed Systems

Springer

Professor Adam Wierzbicki
Polish-Japanese Institute of Information Technology
Ul. Koszykowa 86
02-008 Warsaw
Poland
E-mail: adamw@pjwstk.edu.pl

ISBN 978-3-642-13450-0 e-ISBN 978-3-642-13451-7

DOI 10.1007/978-3-642-13451-7

Studies in Computational Intelligence ISSN 1860-949X

Library of Congress Control Number: 2010927588

Typeset & *Cover Design:* Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

To my wife Małgosia, for all her love, support and patience while I wrote this book.

To my son Tomek, for reminding me that growth takes time.

To my family - without their support and help I would never have finished.

To my students and Ph.D. students, because teaching is the best way of learning.

Preface

This book is an attempt to bring closer the greater vision of the development of Social Informatics. Social Informatics can be defined as *a discipline of informatics that studies how information systems can realize social goals, use social concepts, or become sources of information about social phenomena*. All of these research directions are present in this book: fairness is a social goal; trust is a social concept; and much of this book bases on the study of traces of Internet auctions (used also to drive social simulations) that are a rich source of information about social phenomena.

The book has been written for an audience of graduate students working in the area of informatics and the social sciences, in an attempt to bridge the gap between the two disciplines. Because of this, the book avoids the use of excessive mathematical formalism, especially in Chapter 2 that attempts to summarize the theoretical basis of the two disciplines of trust and fairness management. Readers are usually directed to quoted literature for the purpose of studying mathematical proofs of the cited theorems.

Social Informatics in general, and Trust and Fairness Management in particular, can benefit from the social sciences not just for a theoretical foundation of knowledge about human behavior, but also for a wealth of empirical observations that can be used to justify design choices and motivate new methods or algorithms. For this reason, this book has attempted to emphasize the existence and meaning of such empirical research in the social sciences. Sometimes it is also possible to point out empirical studies from informatics that may be of interest to social scientists. These studies are usually based on data mining and exploration, and can result in hypotheses that may be verified by social experimentation.

The book has based on research supported by two grants of the Polish Ministry of Science and Higher Education: 69/N-SINGAPUR/2007/0 (mTeam) and N N516 4307 33 (uTrust).

Contents

List of Figures

List of Tables

Chapter 1
Introduction

Love all, trust a few,
Do wrong to none.
"All's Well That Ends Well".
William Shakespeare

Information technology today is a vast body of knowledge that has found applications in most areas of human activity and thought. With the proliferation of the Internet and its various applications, the role of information technology in society has become vital and is rapidly increasing in importance. It can be said that the Internet has become intertwined with society; and with each new generation of Internet users, information technology becomes applied not only more widely, but also affects our society and personal lives more deeply. We are witnessing the birth of the real e-society.

For these reasons, is becoming important to start asking questions concerning information technology that were previously been of little meaning. At first, the most important criterion for the evaluation of information technology was performance. With increased adoption of information technology, issues of security and usability became more important for IT users. *Now, as an entire society begins to use the Internet and IT technology, new criteria are becoming necessary for its evaluation.* These criteria are based on the issues that have always been of the highest importance for societies. Among these, a prominent role is played by the two concepts of *trust and fairness.*

Considering trust and fairness in information technology development also raises new challenges for systems research, mathematics, economics, sociology and psychology. The birth and growing maturity of the e-society is leading to the development of new, interdisciplinary areas of scientific inquiry, such as the social aspects of Internet usage or the psychology of computer games. Information technology should benefit from the results of the other disciplines. This is especially important since information technology already largely copies social mechanisms. Examples of such copies are Internet auctions, Internet fora, and social systems using Web2.0 technology. While the social mechanisms that are adopted by information technology may have

A. Wierzbicki: Trust & Fairness in Open, Distributed Sys., SCI 298, pp. 1–10.
springerlink.com

been proven to work in real societies, they often require improvement in the e-society because of the inherent properties of systems created by information technology. In this book, we focus on two such properties: the *openness* and *distribution* of information systems. These two properties have, in our opinion, a fundamental impact on the e-society.

In this book, we consider two areas of modern information technology: trust management and fairness management in open, distributed systems. While these two areas have mostly been considered in separation, this book will aim to demonstrate their interconnections, and to establish a theoretical foundation for the joint consideration of trust and fairness. We shall attempt to show how results from social sciences can be applied to understand and possibly improve trust and fairness management. However, the book takes a systems approach, and presents results from the area of information technology. Interestingly, while some of these results have been inspired by the social sciences or by the mechanisms of real society, other solutions for trust or fairness management seem to be the result of purely technical or mathematical thought. We hope that the contents of the book could also prove interesting to representatives of the social sciences, especially if they are interested in the e-society. In this work, we have attempted to create a basis for common understanding between the social sciences and information technology in the area of trust and fairness management.

1.1 Open Distributed Systems

The concept of *Open Distributed Systems* (*ODS*) is used as a model for many kinds of systems studied in this book. The ODS model is very general, yet it attempts to capture the most significant properties of systems used in the e-society. Using the ODS model also us allows to recognize similarities between solutions used in widely different applications or even to transfer successful mechanisms to new application domains that can also be modeled as ODS.

ODS are an important part of the dynamically developing Knowledge Economy. We can define ODS as systems that are decentralized in the sense that there is not exist a complete centralized control over all system components and functions, and which are open in the sense that agents are able to join and leave the system easily at any time. Examples of ODS include: the IP network, Peer-to-Peer (P2P) overlay networks, ad-hoc networks, grid systems, online auctions, e-commerce systems and virtual organizations (for example, e-businesses) based on Web Services and the Service-Oriented Architecture (SOA). The success of ODS is mainly due to the fact that they create new possibilities for interactions between independent participants (*agents*). This is achieved by lowering the barriers of entry and communications between agents. In short: increasing the openness of the system. However, due to the distributed nature and large scale of ODS, complete, centralized control cannot be ensured. Moreover, the openness of the ODS results in dynamic

changes of ODS structure. ODS dynamics makes it even harder to design effective control mechanisms.

As a result of ODS openness and distribution, agents make decisions under uncertainty: their outcomes depend on the decisions of others. Furthermore, the openness and lack (or weakness) of central control in ODS often makes it impossible to prevent conflicts of interests between agents. In ODS, the question of fairness therefore becomes important. Increasing the fairness of ODS can increase the participation of agents in these systems. Equally important as the fairness of the system, the issue of trust in other agents is of great significants for ODS users. Trust allows these users to participate in the ODS even if they cannot be completely certain of how their outcome will be affected.

To see why ODS are closely related to trust and fairness, consider for a moment a system that is not an ODS. This means one of two things: either the system is closed, by which we will mean that the set of agents that participate in the system is not very large and known in advance; moreover, no (or very few) new agents join the system during its operation. Another possibility why the system is not an ODS is that it has a permanent centralized control over most operations and functions (the two possibilities are not mutually exclusive). In the first case of a closed system, agents will have repeated interactions with each other, and this should constitute a strong incentive to behave fairly, as well as remove the uncertainty caused by dealing with unknown agents. If an agent does not follow the rules of the system, he can most likely be detected and punished, possibly by removal from the system or by a reduction of his access privileges. In the latter case of a system with centralized control, the controller will have knowledge about all agents, and the possibility of using this knowledge to detect and punish misbehavior in a similar manner as in the closed system. This ability of the controller removes the need for trust among the agents, replacing it with the need of strong trust in the controller. As a matter of fact, many centralized systems (such as client-server systems) make use of an a priori assumption of complete trust in the centralized control element.

The above reasoning shows that the definition of an ODS is sometimes fuzzy, because the important thing is the *degree* of openness and lack of centralized control. Often it is quite possible to rely on a completely trusted centralized control, and indeed sometimes this is the only practical solution. Yet, in many cases, there exist important reasons for attempting to reduce the functions and involvement of the centralized controller in the system. Similarly, in some applications it is possible to control the membership of agents (although this is a less frequent approach in Internet-based applications).

While open, distributed systems are a subject within interest of information technology, there are several examples of such systems that are completely non-virtual. In these non-virtual ODS, issues of fairness and trust are none the less important.

1.1.1 Non-virtual Examples of ODS

Many aspects of real society can be modeled as ODS. Consider the problem of shopping for groceries. The groceries market can be considered as an open, distributed system without strong and permanent central control. Buyers encounter the risk of purchasing stale or unpalatable groceries. While they may inspect the goods directly, sometimes this is not enough, as not all flaws may be discovered without tasting or opening the goods. For that reason, buyers often rely on brand in their selection of goods, and select brands that have been tested before. The use of brands is of course a mechanism found on most markets. In Trust Management terms, brand is referred to as reputation, and is one of the most important TM methods. Note that in virtual markets, inspecting the goods is often impossible or very difficult, and reputation is frequently a very important source of information in the selection of sellers or goods. Fairness Management is also required in a groceries market, but it is mostly reduced to ensuring the procedural fairness of transactions. For example, if someone buys stale groceries, the seller is usually expected to return the money. If he does not, it would be theoretically possible to go to court; notice, however, that this is usually impractical, and that the real punishment of the seller is by reducing his brand value. This example immediately shows the interconnection between Trust and Fairness Management in non-virtual ODS. Market mechanisms are also a special method to solve very general distribution problems. Distributional and procedural fairness are therefore implicitly considered by markets.

A completely different example is car traffic. Car traffic can once again be modeled as an open, distributed system. The decisions of drivers are autonomous and can have a large impact on the safety of others. Many diverse mechanisms motivate drivers to act fairly (following the established car traffic laws). The strongest by far is the instinct of self-preservation; yet, in some drivers, this motivation is not always sufficient. A question arises as to why we are willing to depend on other drivers obeying basic traffic laws to some extent. Reputation cannot be applied in this case, since we know nothing about other drivers' past behavior. The degree of trust we have concerning other driver's behavior is obtained, among other things, from the existence of the traffic police. Since the police can sometimes intervene and punish unfair behavior, car drivers can be expected to follow the basic traffic laws. Once again, traffic police function as a mechanism for ensuring procedural fairness, but also establish trust in drivers who obey traffic laws.

In observing the two above examples, it should be evident that many phenomena that occur in a non-virtual society can be modeled as open, distributed systems. Yet, this book will focus on methods of information technology that must be applied for virtual or partially virtual ODS. Due to the increased openness of virtual ODS, many of the mechanisms from real-life cannot be directly applied in virtual ODS. Law enforcement is one such example. Due

to the weakness of international law enforcement, differences in national laws, and the geographical distribution of ODS users, law enforcement is very hard to apply in order to ensure fairness and establish trust in virtual ODS. In the next section we will consider examples of virtual ODS and discuss the special requirements of these systems that lead to the study of trust and fairness in ODS.

1.2 Trust and Fairness Requirements in ODS

The development of virtual ODS and of subsequent special problems, such as Trust and Fairness Management in ODS, is a result of the ongoing innovation processes. Figure 1.1 illustrates how innovation can lead to the development of new distribution problems, and in turn, to the use of new methods to solve these problems. The role of innovation in this process is the creation of new resources or costs that need to be distributed. As a first example, consider radio communication, which introduced the problem of fair distribution of communication frequencies. This problem is now solved by international and national agreements and law, for example by the International Telecommunication Convention[1].

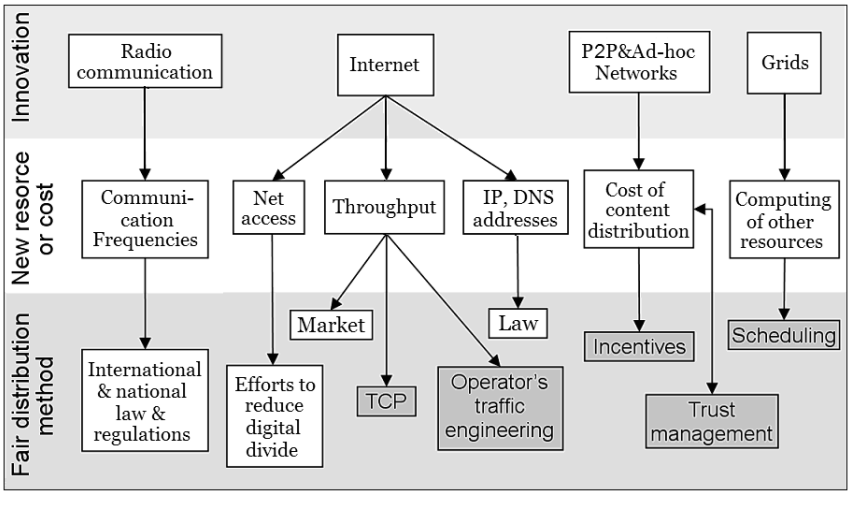


Fig. 1.1 Examples of Fairness Requirements Due to Innovation

[1] This document, dating from January 1st, 1934, established the International Communication Union. It has since been revised many times, and thoroughly changed in 1992, but remains the main source of international law regarding the global distribution of communication frequencies.

The Internet is another example of a rather far-reaching innovation that has, among other things, lead to several distribution and trust management problems. In this discussion, the concept of the Internet is limited to the first four layers of the ISO/OSI stack, excluding Internet applications that are considered separately. A number of distribution problems have become of increased importance due to the widespread adoption of the Internet. Figure 1.1 shows three examples of such problems: the problem of fair network access, the problem of fair network throughput distribution, and the problem of IP and DNS addresses distribution. The first of these is not really a distribution problem, rather; it is a problem of providing fair conditions and opportunities for citizens. A number of government and international aid programs in many countries aim to solve this problem, by reducing the so-called digital divide: providing fair access to computers and the Internet. This subject is especially important in developing countries and for children living in poverty. Fair DNS address distribution is a subject of ongoing legal controversy, while the distribution of IP addresses is sometimes the issue of international politics. Another problem, the issue of fair throughput distribution, has several dimensions. Throughput can be distributed dynamically in the short term between Internet users by the TCP protocol – one of the first examples of a technological solution to fair distribution problems in the Internet. However, network throughput is also controlled by Internet service providers, who can influence its distribution using traffic engineering (such as MPLS technology). The operators can take into account the prices paid by Internet users, but also need to take into consideration the question of fairness.

The Internet also causes a need for Trust Management. A number of security threats with regard to the Internet cannot be solved without trust. The very basics of Internet security, such as the distribution of keys for public-key cryptography, rely on trust relationships; without trust in the certificate authorities of the Public Key Infrastructure (PKI), e-business, e-commerce and e-banking could not function. The example of PKI also demonstrates the need for an improvement of Trust Management in the Internet, as Public Key Infrastructure adoption is slow and often delays the adoption of new Internet applications. Greater flexibility and resilience of Trust Management could improve key distribution, resulting in greater security and faster adoption of new applications.

Among Internet applications, there are numerous examples of fair distribution problems. Figure 1.1 displays two of them: the problem of fair distribution of content distribution cost in P2P networks, and the problem of fair distribution of grid resources. The first problem is solved in a number of ways, among them incentives for participation in cost sharing, and reputation systems (which are a type of trust management). Fair resource distribution in grids is solved using scheduling. These examples all show how technological means can be used to solve fair distribution problems that arise due to innovation. Most of these problems today arise in various types of Open

Distributed Systems, mainly because this is the system architecture of choice for Internet applications.

Figure 1.1 does not exploit the whole spectrum of fairness requirements in ODS. Let us mention the problem of Internet auctions, or the problem of Internet games, both of which are a special kind of fairness problem. These problems are not distribution problems, but rather problems of "fair play". Still, they are an important variety of fairness problems in ODS: the main source of these problems is the possibility of the violation of fair agreements by users who wish to exploit others.

Let us look at some of the areas mentioned in more detail, in order to understand their requirements for trust and fairness.

1.2.1 Network Protocols

Some network protocols, notably the Transfer Control Protocol (TCP), link layer multi-access protocols, need to consider fairness explicitly. One of the goals of these protocols is the solution of distribution problems, like the distribution of throughput by TCP, or the distribution of link bandwidth by the multi-access protocols. Most of the proposed solutions cannot use centralized control, with the exception of some multi-access protocols that use polling or other forms of centralized coordination. Fairness is also an issue for IP routing protocols that control the critical infrastructure of the Internet. IP routers execute the Border Gateway Protocol that takes into account market agreements between Internet service providers and thus solves a distribution problem. Trust is an issue for access control protocols used widely in the Internet. Many of these protocols rely on implicit or explicit trust relationships, and newer protocols also utilize trust negotiation mechanisms.

1.2.2 P2P and Ad-Hoc Networks

One of the recent advances of Internet technology has been the development of the Peer-to-Peer (P2P) computing model that attempts to exploit the resources of Internet hosts (end-systems) to achieve a greater scalability and fault-tolerance than traditional client-server systems. The P2P model is also applicable in circumstances when centralized server infrastructure is not available, such as in wireless ad-hoc networks. P2P and ad-hoc systems have many similarities, as both need to solve routing problems in very dynamic environments. Both P2P and ad-hoc networks are good examples of ODS.

The lack of central control in P2P and ad-hoc networks creates the possibility of exploiting other agents, often referred to as *free-riding*. Combating free-riding is a form of Fairness Management. P2P and ad-hoc systems are also vulnerable to malicious adversaries that can alter the behavior of services. Trust Management can be used to combat such adversaries.

1.2.3 E-commerce

Global e-commerce systems often rely on Trust Management (usually reputation) systems to combat fraud that cannot be prevented using global law enforcement. One area experiencing such problems are popular Internet auctions such as e-Bay. Reputation in these systems is an important capital that can have an impact on prices and revenues. Still, reputation systems used in e-commerce today are vulnerable to whitewashing, discrimination, coalition attacks, Sybil attacks, and a variety of other adversary strategies that will be the subject of chapter 3. Their improvement is therefore of significant importance to the global e-economy, especially when taking into consideration the volume of e-commerce today.

1.2.4 Virtual Organizations and Grids

E-business and e-government are various forms of virtual organization that are often constructed using a set of network protocols referred to as Web Services. These protocols enable the composition of services provided by various organizations into more complex services or workflows. Another example of systems that use Web Services are grids. All systems using Web Services are examples of ODS, although they can have more centralized control than, for example, P2P systems – usually because of a smaller scale. Still, virtual organizations and grids also make use of TM and require FM to solve complex distribution problems. The latter issue is especially important in grids that distribute resources through complex scheduling.

1.2.5 Fair Resource Allocation in Telecommunication Networks

Telecommunication networks are usually managed by a single organization which reduces the need for Trust Management. However, network operators must solve a problem of resource distribution, usually by a special design or traffic engineering in their networks. These distribution problems require a consideration for fairness. The solutions obtained can be applied dynamically using traffic engineering based on MPLS technology.

1.3 Goals of This Book

Many of the solutions that satisfy fairness requirements both in virtual, and non-virtual ODS have similarities in their approach and structure. A schema for such solutions is shown in Figure 1.2. The various kinds of fairness solutions used in ODS rely on fair procedures or negotiations for the establishment of a fair resource or cost distribution or for the establishment of a fair

agreement. Fair agreements or fair distributions are sources of rules of individual fair behavior. Fair procedures can also be the source of such a rule that leads to the emergence of a fair distribution. However, in an ODS, rules of fair behavior may not be followed; users need incentives for following rules, or rules need to be enforced. Another issue is that the rules can be questioned, and that their fairness needs to be validated. Most of the examples of fair technological solutions discussed above are examples of fair procedures: TCP can be thought of as a fair procedure for dynamic throughput distribution; scheduling in a grid system is another example. Examples of fair agreements are the rules of an Internet game, or a transaction that has been made in an Internet auction system.

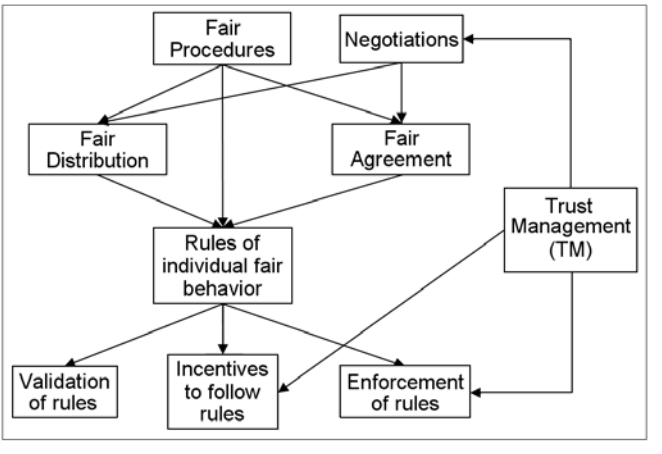


Fig. 1.2 Schema of solutions for satisfying fairness requirements and the role of Trust Management

Trust management can be used to support a variety of solutions for satisfying fairness requirements. Enforcement of and incentives to follow fair rules can be supported by trust management. An example of an incentive is the value of reputation to a user. Examples of rule enforcement using trust management are the use of escrow services in Internet auctions, or the use of trust management in Internet games (see section 4.2). In addition, trust management can also support the process of arriving at a fair distribution or a fair agreement; for example, the reputation of users can be used in the process of an Internet auction, which is an example of a negotiation in an ODS.

The subject of this book will be the various algorithms, system designs, protocols and methods used for Fairness and Trust Management in virtual ODS. Our goal is to emphasize and point out the interconnections between the two areas, as well as to demonstrate the common features of various Trust or Fairness Management approaches. The next chapter of this book will

aim to create a synthesis of the theoretical knowledge of Trust and Fairness Management, based on various disciplines, not just on information technology. This chapter presents a theory of distributional fairness referred to as *theory of equitable optimality* (see section 2.3.2). This theory will be used throughout the book because it provides measures of distributive fairness and methods for improving distributional fairness or even finding solutions that are optimal with respect to distributional fairness. Chapter three is devoted to Trust Management and attempts to present universal methods and algorithms of TM, while also discussing specialized TM applications. Chapter four covers different kinds of Fairness Management, discussing various case studies of application of the theory of equitable optimality introduced in chapter two. These applications have been selected in order to represent different solutions methods, system designs and usage scenarios. Chapters three and four base on the theory presented in chapter two, which is build required for a deeper understanding of the presented methods.

Among the most significant new findings reported in this book is the validation of the *Fairness Emergence Hypothesis* at the end of chapter 4. This hypothesis states that if a trust management system is working well, then improved distributional fairness (in the sense of the theory of equitable optimality) will emerge in the system. This allows to circumvent the difficulty of providing distributional fairness in a completely decentralized system. The validity of the hypothesis also has theoretical significance, because it shows that in systems (or human societies) with no trusted central control, a measure of distributional fairness can be achieved if most agents can trust each other (even if some agents behave unfairly). The book also proposes a new trust management system for Internet auctions (see section 3.5) based on a detailed analysis of a trace of real Internet auction usage. The conclusions of this analysis allow us to define a better way of representing and processing computational trust. A theoretical value of the empirical analysis of Internet auction users is that it confirms another theoretical relationship between the concepts of trust and fairness. In the comments sent to the reputation system of the Internet auction platform, users most commonly report violations of fair procedures that are part of a code of conduct for Internet auctions.

Chapter 2
Theory of Trust and Fairness

Hard models, soft thinking; soft models, hard thinking. Harold Barney, 1978

The subject of this book is an area of research that is in the scope of interests of many disciplines. While the main focus of this book is on information technology, it cannot be denied that sociology, economics, psychology, as well as mathematics (most prominently, game theory) have contributed extensively to the subject. Moreover, research in information technology often based on the results from other disciplines. For this reason, the following chapter will summarize the most important theoretical contributions of the various disciplines, especially with regard to their possible contributions to information technology.

2.1 Basic Concepts of Trust Management and Fairness

The goal of this section is to introduce the most general concepts used throughout the book. The definitions of these concepts will not be stated formally, but rather we shall aim to establish an easier understanding of the terms, which will be considered in more detail (and in some cases, using mathematical formalism) in later parts of the book.

2.1.1 Agent, Encounter, Action and Context

The participants of an ODS will be referred to as *agents*. The interactions of agents in the ODS take place during *encounters* of two or more agents. During an encounter, each agent can choose and execute one or more *actions* (the choice of these actions is usually autonomous). Finally, the concept of context is difficult to define, as it has a different meaning depending on the application. Here, let us state that *context* is *metainformation about the agents, the encounter, or about encounters of third parties*.

A. Wierzbicki: Trust & Fairness in Open, Distributed Sys., SCI 298, pp. 11–69.
springerlink.com

An important consequence of our encounter definition is the conclusion that an agent's decision in the encounter can be the choice of one of the available actions. Contrary to much trust management research, the *decisions made by the agents are therefore not binary.* An agent does not make a decision whether to trust or not to trust; rather, the agent chooses one of the available actions (in some encounters, there may be just two actions available).

2.1.2 Trust Management

Trust Management (*TM*) can be defined as a special method of decision support under uncertainty. The most general and abstract statement of the TM problem is in fact as a decision making problem: an agent considers a situation in which she must make a decision about the choice of action. The outcome of the action is uncertain, as it will depend on the decisions made by other agents. In an ODS, the behavior of agents is autonomous and usually not subject to central control. Moreover, agents often join or leave the system, and the number of agents in the ODS can be very large. Therefore, an agent often encounters other agents previously unknown to her. This is the reason for the uncertainty of the outcome. The aim of Trust Management is to *support the agent in making a decision under uncertainty by establishing trust or distrust in the other agents on whose actions the decision maker's outcome depends.*

2.1.3 Trust, Trustworthiness and Reputation

Trust is a concept derived from the humanities and our own common experience. In fact, its use in information technology can be seen as an instance of anthropomorphic reasoning, as the agents in an ODS are not always humans, yet the concept of trust is still applied to them[1]. There are several known definitions of trust that will be discussed in more detail below. Here, let us define trust (and distrust) in the most abstract manner as a *relation between a trustor and a trustee in a context*. The existence of this relation can be represented as a specific state of the trustor. Both trustor and trustee are agents or groups of agents of the ODS.

Let us here introduce another important aspect of our definitions of trust. While the definition of trust given above should remain valid even if we introduce more detailed and restrictive definitions later, it is important at this point to distinguish between two kinds of trust that will be considered in

[1] Note that in our previous definition of ODS in section 1.1, it was not assumed that agents (if they are not humans) are intelligent or apply reasoning. However, this possibility should not be excluded. An agent can be modeled using, for instance, the Believe-Desire-Intention framework, or some similar formalism of Distributed Artificial Intelligence (DAI) or Multi-Agent System (MAS) theory. Then, the ODS can be considered as an intelligent system.

this book. These are *human trust* and *computational trust*. Throughout the book, the term human trust will refer to a mental state of humans. Human trust has been studied by psychology, sociology, anthropology, economics and other sciences (such as neuroscience), and many of the definitions and observations described later in the book apply to some extent to human trust. However, it should be clear that human trust is a very complex concept that can perhaps never be precisely defined and fully understood. Computational trust, on the other hand, is a term that describes representations of trust used in trust management systems. These representations can be based on trust definitions that could also apply to human trust. Moreover, computational trust is usually processed in a manner that aims to replicate how humans reason about human trust. Finally, the aim of trust management systems is to establish human trust through computational trust, if the users of the TM system are humans. In some cases, the agents that use the TM system are artificial, and then the TM system only provides information in the form of computational trust that is processed by these agents (perhaps using some reasoning rules).

The concept of trust is defined here as a relation. However, a related concept is *trustworthiness*, which is not relational, but is instead a property of an individual agent (or a group of agents). Trustworthiness can be defined as the *objective, context-dependent quality of deserving trust*. In other words, an agent is trustworthy in a context if many other agents can trust him in this context. The concept of trustworthiness can differ depending on how many agents should trust the trustor: we can require that all agents or a majority of agents do so. Frequently, trustworthiness is believed to be a function of an agent's attributes, like the agent's professional standing, external appearance, or other objective properties. However, this is a misconception. The attributes of an agent that are believed to be the basis of trustworthiness are the reason for trust propagation based on similarity (see section 2.2.7). Trustworthiness is an intrinsic property of an agent that can be thought of as a function of the agent's norms or values (if the agent is human). Moreover, it is hard to use the concept of trustworthiness in practice. Almost all data that can be obtained by a trust management system is relational in nature; because of this fact, it is much easier to evaluate or estimate trust than it is to judge the trustworthiness of an agent.

Another concept that can be derived from our common sense, and from economics, is the concept of *reputation*. The distinction between trust and reputation is one of the earliest theoretical achievements of Trust Management research [143]. The most abstract definition of reputation is: *information about the trustee that is available to the trustor and is derived from the history of the trustee's behavior in some contexts*. Note that from this definition it can be seen that reputation can establish human trust (and therefore, in some trust management systems, computational trust is not used at all. These TM systems shall be referred to as *reputation systems*).

2.1.4 Risk and Uncertainty

For the purposes of this book, we shall use a simple definition of risk: *risk is the expected variation of the outcome of an agent in an encounter*. Note that this variation may be calculated precisely only if the agent knows the probabilities of the other agent's behavior (and if the agent's behavior can be modeled probabilistically). Then, risk is determined by the expected deviation from the expected outcome, and is therefore always expressed in the same units as the agent's outcome. There can be many ways of estimating risk, as sometimes the available samples are too small to use simple probabilistic formulae; in credit risk, for example, complex econometric and data mining methods are used to estimate risk. In this chapter, we shall assume that a *risk estimate* is available; however, methods of calculating risk estimates are beyond the scope of the theory of trust.

In many practical cases, there is insufficient information about the encounter and the participating agents to calculate good risk estimates. Instead of the term of risk, we shall use the term of *uncertainty* to denote the *unknown variation of outcome*. The worst-case estimate of risk could be assumed in such a case: if an agent chooses an action in an encounter, then his outcome will depend on the actions of other agents. The worst case estimate of risk is then the difference between the best case outcome and the worst case outcome for the chosen action.

2.1.5 Fairness

After trust management and trust, fairness is the most important concept of this book. It is therefore critical to understand it clearly. Although extensively studied [190], fairness is a complex concept that depends much on cultural values, precedents, and the context of the problem.

Fairness has been studied for a long time in the areas of mathematics and economics, as well as in philosophy, anthropology and other social sciences. It is important to understand that the concept of fairness does not have to use concepts from the area of ethics or morality. Fairness can be defined clearly in a specific context, and further, some concepts of fairness can have a strong mathematical foundation. This statement does not mean that ethical or moral considerations do not have an impact on fairness. On the contrary, ethics and morality establish norms of human conduct that influences fairness. The relationship between these concepts becomes clearer when using the abstract definition of fairness outlined below.

Much of the research on fairness has been carried out in the area of the social sciences, especially social psychology. The results of this research allow us to understand what the preferences of people are regarding fairness, and how people understand fair behavior. Research on fairness in the social sciences will be reviewed in section 2.3.4. Interestingly, much of the research in this area has been influenced by the seminal work of Deutsch, who is also the

author of one of the basic psychological theories of trust [38, 40]. To begin our discussion of fairness, let us begin with three general kinds of fairness judgements identified by social psychology [166]: *distributive fairness*, *procedural fairness* and *retributive fairness*. This book focuses mostly on distributive fairness. Distributive fairness is usually related to the question of distribution of goods, resources or costs, be it kidneys for transplantation, parliament mandates, or the costs of water and electricity. The goal of distributive fairness is to find a distribution of goods that is perceived as fair by the agents concerned. Procedural fairness focuses on the perceived fairness of procedures leading to outcomes, while retributive fairness is concerned with rule violation and the severity of sanctions for norm-breaking behavior. It is possible to think of distributive fairness as a special kind of procedural fairness. If a distribution problem can be solved fairly, then a fair procedure would require all agents to take a fair share of the distributed goods or cost. Procedural fairness, however, is also applied in the case where a fair solution cannot be found beforehand or cannot be agreed upon. Both distributional fairness and procedural fairness aim to find fair solutions of distribution problems.

The most abstract definition of fairness used in this book is therefore as follows [190]. *Fairness* means the *satisfaction of justified expectations of agents that participate in the system, according to rules that apply in a specific context based on reason and precedent*[2]. This general definition applies to distributive, procedural or retributive fairness.

It is also useful to distinguish between *individual fairness* and *system fairness*. By individual fairness we shall mean the property of an individual agent's behavior. We shall say that the agent behaves fairly if he obeys certain rules of behavior that apply in a specific context (the agent behaves unfairly if he disobeys any of these rules). These rules could be contracts, agreements or norms that govern the behavior of all agents in a system. If all agents that participate in an encounter behave fairly, then the distribution of goods (or costs) of the agents in this encounter will be fair in the sense of procedural fairness. System fairness can be defined as the ability of a system to support the individual fairness of agents in the system. In the context of distributional fairness, system fairness can also mean the ability of a system to support the distributional fairness of all agents' participation in goods or costs.

Note that the rules that determine individual fairness can be derived from the rules that determine distributive fairness: an individual agent should not take more goods or incur less costs than are assigned to him using the rules of distributive fairness. On the other hand, if a notion of fair distribution is not available or not enforceable, then individual fairness can be dictated by procedural fairness.

There can be many kinds of distribution problems. An important distinction is between the distribution of *divisible goods* (like money, or network bandwidth) and *indivisible goods* (like kidneys for transplantation or

[2] According to the Oxford English Dictionary, the word "fair" means: equitably, honestly, impartially, justly; according to rule.

mandates in an election). Another kind of distinction is between the distribution of *heterogenous* and *homogenous* goods. In many cases, such as markets, the distribution problem is reduced to problems of exchange of some goods for other goods (usually money). Distributive fairness in markets is achieved by mechanisms that allow the determination of a fair rate of exchange of these goods (a fair price). Then, the considered problem is of heterogenous goods distribution.

It can be seen from the above examples that the problem of distributive fairness is extremely general and has a wide range of instances, especially in the non-virtual society. For an excellent overview of distributive fairness methods and applications, see [190]. In this book, we shall focus on distributive fairness in ODS.

2.1.6 Equity, Justice, Reciprocity, Altruism

The concept of *equity* is sometimes identified with that of distributive fairness. However, in this book, equity will have a stronger meaning than that of distributive fairness. To intuitively understand the difference between the two, consider a problem of distributing two kinds of goods between agents. The first kind of good is the profit made by a group of agents who worked on a project together (for example, the employees of a company). It is clear that *a fair distribution* of this profit *that aims to minimize the differences in the shares of equally entitled agents* is desirable. We shall refer to such a distribution as not only fair, but also equitable. Equitable distribution relies on the ability of determining that certain agents are equally entitled to a share of goods or costs, by examining certain properties or attributes of the agents themselves. If all agents can be compared, but are not equally entitled, the concept of equity can be extended. Agents should then receive shares according to their entitlement. A popular notion of equity is the notion of a distribution that is proportional to the contributions of agents. This notion, according to Deutsch [38], is one of the three basic psychological expectations of a just distribution. (The other two psychological expectations are equality and distribution according to need.) Note, however, that the expectation of a proportional distribution may be intuitive only in certain cultures, as for example traditional Jewish law does not propose proportional distribution [190].

If the participating agents cannot be compared or if the minimization of differences between their shares is not a goal, we can still speak of distributive fairness. However, in such a case we shall not refer to the distribution as equitable.

Equity is a concept that is narrower than *social justice* [137]. Social justice concerns itself with the total welfare of agents in a society, considering all (even unmeasurable) goods that may influence an agent's welfare (including the right to public services such as education, health care and safety, or social attitudes that determine the self-esteem of people).

Finally, the concepts of *reciprocity* and *altruism* can both be considered as special cases of individual fairness. The rules that govern the behavior of individual agents can be determined by the context of the encounter. In some contexts, fair individual behavior is reciprocal behavior. In other contexts, altruistic behavior is expected of agents (for example in the case of a parent caring for a child). Therefore, most of the reasoning that applies to individual fairness would also apply to agents that behave reciprocally or altruistically, in an appropriate context. On the other hand, sometimes reciprocity and altruism are not individually fair behavior (a salesperson would not behave fair if he altruistically gave goods away for free). Note that in this book, the term altruism will be applied only to the situation when an agent prefers another agent's welfare to his own. An act is said to be altruistic if it is costly to perform but confers a benefit on another agent [182, 165]. Altruism is also widely used (especially in game-theory and economics) to mean "un-selfishness" (in constant sum games, the two notions are equivalent). In the terminology used in this book, unselfish behavior can be any fair behavior, while altruism has a stronger meaning (and is also much more rare [25].

2.2 Theory of Trust

This section introduces the most important findings of previous research on the concept of trust. Most of this research concerned human trust, without explicitly making the distinction between human and computational trust. It is often assumed that computational trust should model human trust as closely as possible, and therefore all research on the subject of human trust would also apply to computational trust. In chapter 3 we shall show that could also behave differently from human trust, and that trust management functions can be carried out even without using computational trust. Still, in many cases it is desirable that computational trust should be used and should model human trust faithfully. Therefore, research on human trust has a high relevance for trust management in information technology, especially if it can be empirically verified.

For the above reasons, this section focuses on human trust. Examples of computational trust are sometimes used to better demonstrate introduced definitions, yet it would be a mistake to consider these examples as instances of how humans think about trust. It is likely that any formula used to calculate computational trust is to some extent inappropriate for human trust, and computational trust can only approximate human trust. More sophisticated methods to calculate computational trust and, more interestingly, ways of using computational trust will be the subject of chapter 3.

This section presents the most important and most commonly used definitions of the concept of trust, and gives a broad overview of the research literature on human trust. However, it should be noted that the list of covered definitions may not be complete; in the social sciences, research on trust is an active area, and new developments can be expected. Another goal of

this section is to present the definitions in a way that will allow the reader to understand the relationships between them, and to show the logical relations between the theory of trust and of fairness. A final goal is to present empirical evidence concerning the use of trust and reputation by humans, and empirical research that can falsify the theory of trust.

2.2.1 Expectancy Trust

According to this definition, *human trust is a subjective, context-dependent expectation that the trustee will choose a specific action in the encounter*. This definition is close to definitions proposed in literature, such as [53] and [118].

An example of expectancy trust is when a buyer on an Internet auction pays for a product in advance (this method is preferred by sellers). The buyer then expects that the seller will send him the purchased item. Another example is Peer-to-Peer file sharing, where a peer is expected to reciprocate for data downloaded from another peer. In grids, agents expect that their jobs will be scheduled fairly by other agents who compute these jobs on their own machines. An agent in a grid can also put foreign jobs at the end of the queue.

Note that in order for this definition to apply, it is not necessary that a human should explicitly calculate and compare the probabilities of choosing an action by the trustee. It would suffice that the trustor would be able to choose a particular action and expect that the trustee will also choose this action. Note also, that if the trustor is able to form such an expectation about all other agents in the encounter, then it will follow that the trustor will be able to choose his own best action and to form an expectation about his own outcome in the encounter.

Apart from support for this definition in much literature in the social sciences, expectancy trust has also received some recent empirical support. The expectation is formed based on information that is relevant to the future outcome of the encounter. In order to verify whether humans form an expectation about the outcome, it is necessary to observe their behavior when the information provided is varied. If the behavior varies, it can be concluded that an expectation is formed and is affecting behavior (under the ceteris paribus assumption that no other factors impact behavior). In [10], the finding that people form expectations based on information about trustees is verified. Barr used an economic investment experiment conducted in small communities in Zimbabwe, and tried to verify whether people acted based on expectations about the trustee in the absence of any other mechanism that would facilitate cooperation (this included the absence of reputation: players were paired into anonymous pairs, and interactions were not repeated). In the experiment, both trustor and trustee came from the same community. The experiment was designed in such a way that the trustor's decision depended on the expected payoff received due to the decision of the trustee, and on the variation of this payoff. If the players expected a higher payoff, they invested more. If the players expected a high variation in payoffs, they usually

invested less. Thus, the findings of Barr give empirical support to the theory that human trust is an expectation. The expectation in the experiment could only be formed based on the experiment setup and based on the knowledge of the participants about their communities.

2.2.1.1 Fairness and Expectancy Trust

Recall that distributional fairness has been defined as satisfaction of justified expectations of agents. If we agree on defining human trust as an expectation, then the concepts of trust and fairness can be logically related. If an agent A trusts another agent B in an encounter (A expects that B will choose a certain action), and B behaves fairly (hence, satisfies the justified expectations of A), then the expectations of A will be satisfied if they were justified. Let us, for the moment, assume that the context of an encounter includes rules that enable agents to verify whether their expectations are justified, and that A would not form expectations that are not justified. If that is the case, then *the trustor can trust the trustee if the trustee will behave fairly*. (Note that this statement uses the definitions of trust and fairness introduced above, yet is also completely in with our common sense).

However, this statement relies on an important assumption about the agents. Not only must the rules of justified behavior be known to all agents that participate in an encounter, but also agents must not form expectations that violate these rules. This assumption may not always be simple to make. Consider a case when two accomplices form a conspiracy to cheat an insurance company. To this end, one of the accomplices drives a car and hits the other accomplice on a pedestrian crossing. The other accomplice fakes an extensive injury and receives considerable compensation from the insurance company that he shares with his accomplice. In this case, the generally accepted rules of behavior (norms) are clearly violated; not only that, but one of the accomplices forms an expectation that the behavior of the other accomplice will violate these norms. However, this example does not invalidate our reasoning, but merely stresses the importance of a context to the consideration of trust and fairness. In this case, the two accomplices have formed their own rules that defined fair behavior in the encounter, and while these rules violated generally acceptable norms (and therefore, changed the encounter context), the trust and fairness of the agents still agreed with our reasoning. Our reasoning here is based on a rationality assumption about the agents; forming expectations that are contrary to the rules of fair behavior can only be possible due to incorrect or incomplete information.

Still, the reasoning presented here does not imply that trust can only be formed if the rules of fair behavior are known. For other, possibly valid definitions of trust, the relationship between trust and fairness is not obvious, and it is possible for humans to form trusting decisions even in situations when rules of fair behavior are unknown or unclear. On the other hand, observation of comments in feedbacks posted by users of Internet auction platforms gives

empirical support to the hypothesis that trust and fairness are indeed related. Users of Internet auctions use the comments in nonpositive feedbacks to point out violations of procedural rules that are part of a code of conduct for buyers and sellers on Internet auctions. For more details, see section 3.5.2.2.

2.2.1.2 Reputation and Expectancy Trust

We have chosen the usual definition of reputation as information about the trustee that is based on a history of the trustee's behavior in context. This definition can be refined towards a probabilistic model. If the complete and correct history of the trustee's behavior is available, and if the trustee's behavior is modeled probabilistically, then we can calculate an empirical probability that the trustee will choose a particular action in the given context (recall that context is metainformation about an encounter, and therefore also includes the set of available actions. The complete history includes all of the trustee's choices of actions from this set.).

Reputation could, as in [118], be defined as these empirical probabilities. Provided that these probabilities are available, it is certainly possible to form an expectation about the trustee's behavior (for example, to expect that the trustee will choose the action that has the highest probability). Therefore, it can be seen that such reputation can directly establish expectancy trust.

There are a few problems with this definition of reputation. The most significant is that the history of behavior may be incorrect. This can be due to several reasons – the available information may be deliberately falsified, or the information about encounter contexts may be incorrect. The available information could also be incomplete, leading to a biased estimation of the probability of the trustee's behavior. Finally, a simple probabilistic model of the trustee's behavior may be inadequate; for example, it may be necessary to model the trustee's behavior using a stochastic process.

Also, it should be clear that the availability of reputation is not a necessary, but may be a sufficient condition for forming expectancy trust. Humans can expect behavior even if information about reputation is not available (recall the example of car traffic discussed in section 1).

2.2.2 Reputation

Reputation, like trust, is subjective and context-dependent. From the definition of reputation, it follows that reputation can only be influenced by the history of the trustee's behavior. A detailed classification of types of reputation, following [118], is shown in figure 2.1. (In this classification, we have chosen to remove the category of prior-derived reputation that has been used by Mui, as it seems to conflict with the chosen definition of reputation as dependent only on history of behavior. This does not mean that trust cannot be prior-derived.)

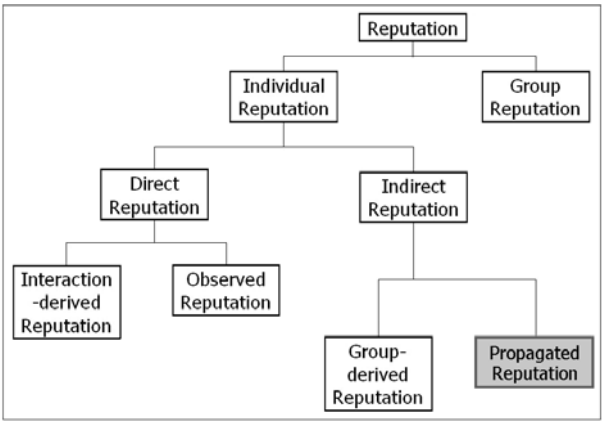


Fig. 2.1 A classification of types of reputation based on

Examples of reputation include the simple reputation used in Internet auctions, where each buyer can read the feedback about a given seller and make his decision on whether to trust that seller. Another example is reputation in Peer-to-Peer systems, where each agent has a certain amount of information about the past behavior of other agents that can be used to calculate a measure of reputation.

Note here that in the literature there exists a distinction between *local* and *global* reputation. Local reputation is equivalent to the definition of reputation as used here: it is subjective with respect to the trustor. On the other hand, global reputation is not subjective to the trustor, although it can be context-dependent. Global reputation is related to trustworthiness, not to trust (see Section 2.1), and is therefore not discussed further. We shall return to this subject in section 3.2, where we will introduce a model of a universal TM system that is capable of calculating both local and global reputation.

The most important distinction used in the presented classification is between direct and indirect reputation. Direct reputation is based on information about the history of an agent's behavior that is directly available to another agent. Reputation based on first-hand experience, for example, is in this category (referred to as interaction-derived reputation by Mui). Another type of direct reputation is observed reputation that can be obtained if an agent can observe interactions of other agents without participating in them. Indirect reputation is based on information obtained through a third party that cannot be directly verified. Reputation systems used by online auctions or other e-commerce enterprizes are in this category, referred to as propagated reputation. Group-derived reputation, as the name suggests, is a reputation of an individual that is derived from the history of behavior of a group. The distinction between direct and indirect reputation stresses the difference between using verifiable information and information that cannot

be verified. The latter approach, most common in reputation systems used today in online auctions, is especially vulnerable.

Any kind of reputation system is vulnerable to so-called *first-time cheating*. This phenomenon occurs when an agent makes a sudden change in behavior. For example, it is possible for a seller on e-Bay to achieve a high reputation, but then to cheat for the first time. Reputation cannot predict this behavior, since, by definition, it is based on behavior history from a time period before the change in behavior.

Indirect and propagated reputation is further vulnerable to adversaries that give incorrect reports about the behavior of agents. While a typical indirect reputation system uses the majority principle to determine reputation, and is therefore resistent to a small number of incorrect reports, adversaries can form *coalitions* in order to increase the impact of their incorrect reports. Indirect reputation systems are not resistant to this behavior.

A final, important vulnerability of reputation systems is vulnerability to *discrimination*. A discriminating adversary will behave fairly towards a majority of agents, but will consistently cheat a minority of agents. By doing so, the adversary should be able to maintain a high reputation, but can get away with cheating a small subset of encountered agents. This type of adversary has been described by [34], who suggested making agents anonymous could prevent their discrimination. However, this approach cannot be used in all cases, and reputation systems are therefore still vulnerable to discrimination.

A more systematic description of adversaries will be given in section 3.3.3.2.

Reputation is a well studied concept, and strong evidence exists concerning the use of reputation by human agents. This evidence comes first from economics, where the concept of reputation is well known. Not only is reputation the subject of study in marketing research; economists have studied here, for example, how reputation impacts the prices of goods and sales volume, also in online systems [139, 140, 41, 62].

For a description of the use of reputation systems in e-commerce, see section 3.3.1.

Reputation has also been considered using game theory. Economists and game-theorists have linked reputation to the existence of cooperative equilibria in repeated games such as the iterated Prisoner's Dilemma. The existence of such equilibria was proven in the 1970's [51], but was postulated since the 1950's in the so-called Folk Theorem [52]. Since then, cooperative equilibria have been observed using simulation in the iterated Prisoner's Dilemma in the famous work of Axelrod [7], and further research in game theory has extended the existence proofs to other types of games [130, 70, 71]. New strategies for the iterated Prisoner's Dilemma have been formed that make use of reputation (Axelrod explicitly ruled out the use of reputation in his definition of the game). The simplest and one of the most effective strategies is Reputation Tit-for-Tat [134], where the reputation of the partner is taken into account in the first move, in order to decide whether to cooperate or defect. In the next moves, Tit-for-Tat is played.

Reputation has also been of interest to evolutionary biologists [134, 27] and anthropologists. In these areas of research, the term "altruism" is often used to describe cooperative behavior towards unrelated strangers [7, 61]. According to evolutionary anthropology, cooperation should be rare and limited to next of kin and sexual partners. Up to recent times, the dominant paradigm in this area of research has been that of rational, selfish behavior. However, recent research has demonstrated that this assumption does not apply to most social groups in a cross-cultural study [61], and that humans have social preferences that support large-scale cooperation. These preferences include concerns for fairness and inequality aversion [45].

For a comprehensive overview of modern reputation systems, see [144]. Section 3.6 contains overviews of related work on using reputation in information technology and computer science.

From this extended look at the theory of reputation it can be seen that there is more empirical research on the subject of reputation than on the subject of trust. Also, it can be seen that in many cases, the concept of trust has been replaced by the concept of reputation in this research. This approach can be seen as a mental short-cut: if trust can be established through reputation, then its existence can be overlooked by empirical research that links reputation directly to its observable changes in behavior. Our example of car traffic demonstrates that trust can also be established by other means. However, sometimes the question of whether trust has been established is doubtful when we are using another definition of trust: that of dependency trust.

2.2.3 Dependency Trust

Dependency trust can be defined as *the subjective, context-dependent extent to which the trustor is willing to depend on the trustee in a situation of uncertainty*. This definition is an adaptation of dependency trust definitions proposed by [67] and [108]. The trust theory of [37] also supports this definition. A similar definition has been used in sociology [158, 157]: trust has been defined there as the attitude that allows the trustor to accept a bet on the trustee's behavior. Accepting such a bet would mean a willingness to depend on the trustee.

The example of Internet auctions can also be used for dependency trust. When a buyer on an auction pays for an item in advance, he expects that this item will be sent to him; however, he also depends on the seller to send the item, and this dependency is high and can be adequately measured by the amount of money paid to the seller. In Peer-to-Peer file sharing, on the other hand, a peer expects that his partner will reciprocate by sending him data in exchange for downloaded data. However, the dependency of the peer is smaller, because there also exist other peers that can send him the required pieces of a file.

Dependency imposes a different condition on human behavior than expectancy trust. The reason why this is so is the following: consider a situation where there is a high expectation that the trustee will behave in a certain manner (choose a specific action in the encounter). Because of this action, the expected outcome of the trustor may also be high. However, the probabilities that the trustee could choose another action could be nonzero, and the difference in the utilities of the trustor when the other action is chosen by the trustee could be high. Consequently, the trustor could face a high risk in this encounter. Because of this, he may choose to not interact with the trustee in the encounter, even though expectancy trust would indicate that he should do so. To give a more concrete example: if a buyer considers an auction on e-Bay where he can buy an expensive car at a good price, and the seller has a high reputation, the buyer may nevertheless choose not to buy the car because of the risk that the car may be damaged, or not delivered at all. This high risk is due to the high price of the car, and not to the small probability of this occurrence.

The two definitions of human trust are not necessarily contradictory (rather, they can be seen as complementary). The premise that a particular concept can be defined only in one way may make sense in the exact sciences or mathematics, but does not necessarily apply to humans, and it is human trust that we are trying to define. It should be possible for humans to use both kinds of trust simultaneously, considering perhaps only expectancy trust if the risk is low, and requiring dependency trust when the risk is high. Following this reasoning, it is not necessary for computational trust to agree with only one of the presented definitions; Trust Management systems could use more than one kind of computational trust. However, the argument behind introducing dependency trust points out that TM systems must also consider the risk in an encounter.

Dependency trust is supported by findings of sociological and psychological research [59, 66]: this research has found that the more frequently an agent is be able to trust others, the more likely he/she is to take risks. Dependency trust emphasizes and explains the relationship between trust and risk, leading us to an attempt to rephrase the definition of dependency trust in terms of risk.

The relationship between dependency trust and risk [68] has led to a common misunderstanding. Since dependency trust is required in a situation where the risk is high, it has been argued that if risk can be reduced, then dependency trust would also be reduced. However, this argument is fallacious: it is *the need for dependency trust* that is reduced if we can reduce risk. *Dependency trust can still be high in a low-risk encounter. On the other hand, it may be true that if the risk in encounters between a trustor and a trustee is always low, then dependency trust may not develop.* The reason for this is that the trustor can only receive proof that can establish high dependency trust

in an encounter that has a high risk[3]. If this encounter is followed by low-risk encounters, the already established dependency trust may still be high.

This argument brings us to a variation of the definition of dependency trust that can perhaps be more computationally useful.

2.2.3.1 Trust as a Tolerance of Risk

Dependency trust can also be seen as a *tolerance for risk*. Risk is the expected variation of the trustor's outcome. The definition of dependency trust as an "extent to which the trustor is willing to depend" points to using units of the trustor's own outcome to express dependency trust (as well as risk). These units could be, for example, amounts of money, as in the case of a car buyer on e-Bay. Moreover, looking at dependency trust as a tolerance of risk points to a rule that can be applied to the use of dependency trust in decision making. If the trustor's risk in an encounter can be represented as an interval on the scale of trustor's outcome (around the expected outcome of the trustor), then so can dependency trust. And if the interval of dependency trust is larger than (includes) the interval of risk, then the dependency trust of the trustor is sufficient for him to make a decision to interact with the trustee. On the other hand, if the interval of dependency trust is smaller than the risk, then the risk in the encounter cannot be tolerated by the trustor.

Defining dependency trust as a tolerance of risk allows us to explain the decision making process of the trustor. If a trustor is involved in an encounter with an agent, he will consider the actions that he can choose. In some encounters, one of these actions is to do nothing (avoid the encounter) – for example, in the example of buying a car on e-Bay, one of the available actions is to not buy the car. When the trustor considers his choices, he will calculate the risk that is associated with each action. This risk is due to the choices that any other agent in the encounter can make (considered separately). Next, the trustor will choose from among the actions for which the risk due to any trustee's behaviors is smaller than the dependency trust in that trustee.

Figure 2.2 illustrates an example of using dependency trust. The trustor has three actions available: $\{A_1, A_2, A_3\}$. On the y axis, the dependency trust and risk due to the actions of the trustee is shown. The dependency trust is the same regardless of the action chosen by the trustor (although in the figure, the position of the interval changes because it is centered around the

[3] Note also that it has been argued [44] that trust management provides information that reduces uncertainty, and therefore may contribute to the decrease of trust. This conclusion need not apply if trust management is used in high-risk encounters: the information provided by trust management may be used to establish trust, but not to reduce risk (most reputation systems are in that category). Therefore, dependency trust can still be established and increased even if trust management is used; although some trust management methods (like escrow) indeed contribute to the reduction of risk.

mean outcome for each chosen action). The risk depends on the action chosen by the trustor. The figure demonstrates that when using dependency trust, one of the actions (A_3) can be excluded from consideration (note that this action could be chosen based on expectancy trust alone). Among the two other actions, the choice could depend on risk and on the expected outcome. Also, expectancy trust can be used in such a case to further support decision making.

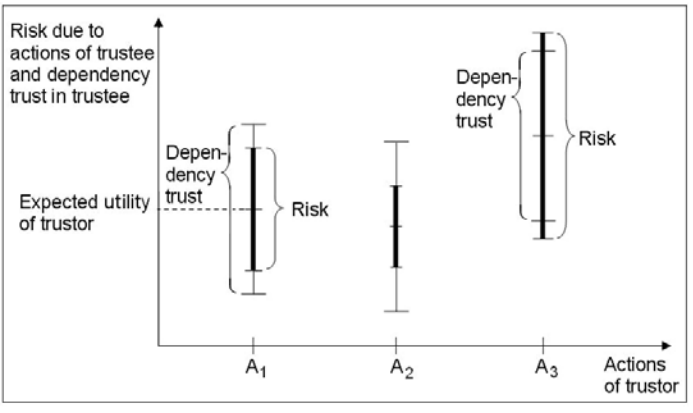


Fig. 2.2 A trustor's decision making based on dependency trust and risk

To further understand dependency trust, let us give a simple example of how it could be calculated. Consider a history of a seller's transactions in an electronic marketplace. Assume that with regard to each transaction, the following information is available: the price paid by the buyer and an observation of the seller's behavior that can be binary: either the transaction has been satisfactorily concluded by the seller, or it was not. (This is clearly a strong simplifying assumption; for further discussion of computational dependency trust, see section 3.5.) From this history, dependency trust of the seller can be calculated as follows: the sum of prices of satisfactorily concluded transactions decreased by the sum of prices of unsatisfactory transactions is then divided by the number of all transactions. Note that such a value could be negative. While this may be a very simplified way of calculating computational dependency trust, it could be directly compared to an estimate of risk (see 2.1.4).

2.2.4 Cognitive and Affective Trust

Research on human trust has always struggled with the criticism that trust is too irrational and subjective to be understood. Establishing the rational basis for human trust is therefore essential for trust management; for, if trust

cannot be treated rationally, there is no way to support its establishment through information technology.

Following the work of [92, 111], we identify two main types of human trust: affective and cognitive trust. This distinction stresses the fact that trust has both an emotional and a rational side. Social scientists have established that affective trust is established through close interpersonal relationships, and is increased by evidence of care and concern between agents. On the other hand, cognitive trust is based on information, and can be established towards strangers. Therefore, cognitive human trust is the main focus of interest of trust management systems.

Figure 2.3 shows the relationship between computational trust and human cognitive trust. The arrows in the figure represent a relation of influence. Proofs of any type can be used by trust management systems to calculate reputation or cognitive trust. These, in turn, can influence the cognitive human trust of an agent. On the other hand, proofs represent information that can also be used by humans directly, and can establish affective trust as well as cognitive trust.

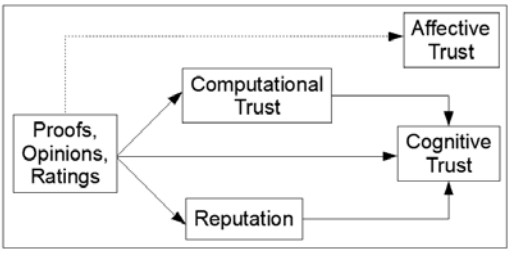


Fig. 2.3 Relationship of Proofs, Reputation, Computational, Cognitive and Affective Trust

Trust management systems, so far, do not use affective trust nor support its formation. In a way, this is practical, as trust management systems aim mainly to establish trust in strangers. A trust management system that would aim to monitor and support a close interpersonal relationship could take into account affective trust. However, since this type of trust is emotional in nature, supporting it would probably require assessing the emotional state of human agents (which is possible using information technology today).

2.2.5 Distrust

If we agree that humans use trust when making decisions under uncertainty, it follows that in some situations trust should actually prevent humans from wrongly relying on others. In natural language, the term distrust is used to

denote this form of trust. We shall attempt here to give a logical definition to this term.

First, note that expectancy trust is general enough to mean distrust as well as trust. Expectancy trust merely allows the trustor to expect that the trustee will carry out a specific action in an encounter. Nothing in the definition suggests that this action would lead to a better, or worse outcome of the trustor. It is therefore quite possible for expectancy trust to be expectancy distrust. In order to avoid confusion, the concept of expectancy trust can be divided into two more specific concepts: that of *positive expectancy trust* and *negative expectancy trust (or distrust). Positive expectancy trust can be defined as an expectation of the trustor that the trustee will choose an action in the encounter with the goal of improving the trustor's outcome. Negative expectancy trust is the expectation of the trustor that the trustee will choose an action in the encounter with the goal of worsening the trustor's outcome.*

These definitions seem ambiguous in the sense that they do not indicate what action would be chosen. However, as we are trying to define human trust, we must not forget that humans are capable of reasoning. Positive expectancy trust may be an expectation about the goals (or values) of the trustee, but if the trustee has a complete knowledge and free choice of all actions in the encounter, the trustor can expect that the trustee will choose the action that maximizes the outcome of the trustor. If the choices are constrained or incomplete, then the trustee will still choose the best possible action from the trustor's point of view.

Of course, it is also possible for humans to use multi-criteria reasoning, and to consider the trustee capable of choosing actions by taking into account both the interests of the trustor (positively or negatively), and his own. It would therefore be possible to define further refinements of expectancy trust that are all expectations about the behavior of the trustee that are based on a model of the trustee's reasoning.

The proposed definition of distrust using expectancy trust shows that distrust can be a negative version of trust. In much trust management research, computational distrust is represented in such a way, using negative units of trust. However, this need not always be the case, for example when dependency trust is used.

The concept of distrust is still not clear when using dependency trust. Let us here choose the definition of dependency trust as a tolerance of risk. In other words, dependency trust is the largest amount of risk that can be tolerated by the trustor and that is due to the behavior of the trustee. This version of the definition of dependency trust can be used for distrust, as well. If an agent is distrusted, then the trustor's dependency trust in that agent will be very small, or even zero; and therefore the trustee will avoid encounters with that agent. If the level of risk in a larger population of agents can be established as a reference, then we could speak of positive dependency trust if dependency trust is larger than the risk in the population of agents, or of negative dependency trust (distrust) in the other case.

2.2.6 Credibility Trust

A concept that is often related to trust, and has important applications in trust management, is credibility. For the purpose of this book, *credibility* will be defined as the *quality of being believable*. *Credibility trust* will be defined as *the trust in the credibility of the trustee*.

Using the concept of encounter context, it is possible to show that credibility trust is indeed a special kind of trust. Consider agents who meet in the ODS solely for the purpose of exchanging information. In other words, during an encounter of these agents, the only action that can be taken is giving information (true, false, or partially true) to other agents. The context of this encounter is information sharing. After receiving new information, an agent will have some possibility to evaluate its correctness (perhaps not immediately, but during a period of time after the encounter). The trust in the agent that has given this information to the trustor in the context of information sharing is credibility trust. Based on that trust, the trustor may or may not accept information from trustees in information sharing encounters. Note that the trustor's outcome may depend vitally on the received information, and the information sharing encounter is associated with uncertainty about the information's correctness.

A similar approach to credibility trust has been adopted by [118]. In his work, Mui has used reputation of agents in the context of giving recommendations. Agents could request recommendations even when they already had their own opinion on the subject of recommendation, solely for the purpose of evaluating other agent's credibility and establishing credibility trust or distrust. Next, credibility trust was used to evaluate reports received from agents that could impact an other agent's reputations. This property of computational trust as used by Mui reflects a property of human trust that has raised significant theoretical difficulties. While it is useful to distinguish between encounter contexts and it is possible to give examples of human trust in different contexts that are not related, it is also sometimes possible to use trust from different contexts simultaneously. This brings us to the subject of reasoning about human trust.

2.2.7 Human Trust Propagation

Most researchers of human trust describe methods of human reasoning about trust. Reasoning is usually applied in order to extend the trust relation to new agents. This kind of reasoning is, in fact, used for one of the main purposes of trust: establishing trust in strangers. Without *reasoning about trust that leads to an extension of the trust relation*, it would not be possible for trustors to have trust in strangers. In this book, such reasoning about trust will be referred to as *trust propagation*.

An example of trust propagation is the use of credibility trust to establish trust in a different context. Consider a trustor, A, that receives information

from a third agent, C, about C's trust in the trustee, B. The trustor has some credibility trust (or distrust) in C. Based on that credibility trust and on the information received, the trustor can adjust his trust in B in a completely different context. Note that before the information was received, A may have had no trust about B (the trust relation has been extended). This example shows two phenomena: that of *transitive trust propagation* and that of *composition of different contexts*. The context of information sharing is an example of a context that usually can be composed with other contexts to establish trust. The reasoning applied here is based on the principle of transitivity: the trust relation between A and C has been transitively composed with the trust relation between C and B, in order to create or change the trust relation between A and B.

To give a simple example of transitivity, consider a businessman who recommends an employee to his colleague. Based on his trust in the recommender, the trustor may trust (or distrust) the recommended employee. Note that the recommendation is more likely to be accepted if the recommender has had first-hand experience with the employee. On the other hand, if the recommendation is passed on from another source without first-hand experience, it is less likely to be accepted. This observation points out that transitive human trust propagation is likely to be constrained, and should not be thought of as a simple transitivity of the trust relation.

It also should be clear that there exists contexts of trust where transitive propagation does not apply. For example, consider a situation where an agent A often borrows books from his friend, B. Based on that observation, it can be concluded that A trusts B to lend him books that interest A. Assume that B also often borrows books from C. The same conclusion about B's trust that C's books are interesting to B applies. The context of that trust relationship is personalized: the trust concerns that received books are interesting to the trustor. However, based on that relation it is hard to conclude whether A should trust that books from C would be interesting to A, if received directly and not borrowed and read first by B.

Another, simple and common type of reasoning about human trust is *reflexive propagation*. This kind of propagation is based on the observation that the more you trust another person, the greater the likelihood that the person will trust you. This finding has been supported by sociological research [59].

A third kind of propagation is *similarity propagation*. This kind of propagation applies if two agents are part of another relation, called a *similarity relation*. Similarity propagation is shown in Figure 2.4. The existence of a similarity relation between agents A and B implies the creation of two new trust relations: A trusts B and B trusts A. The similarity relation can be variously defined. It is usually undirected, and can be based on agent attributes such as ethnic background, institutional membership, professional background or others. The similarity relation can also be defined as structural or topological similarity in some social network (for example, the network of acquaintance). A special kind of similarity relation is defined as topological similarity in the

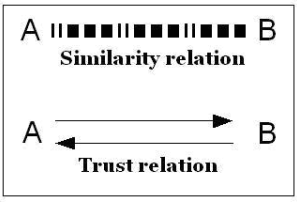


Fig. 2.4 Similarity propagation of trust

trust network itself. Agents A and B are similar if they trust a large set of same trustees. If B also trusts another trustee, C, that A has no relation with, then it could follow that A will also trust C.

Similarity propagation is often combined with transitive propagation. Figure 2.5 B will trust C because they are similar, and since B trusts C and A trusts B, then A will trust C because of transitive propagation. This kind of propagation can also be referred to as a special type of similarity propagation, if the new trust relations between B and C are not explicitly established, but the trust between A and C is established. Similarity propagation can be used instead of transitive propagation especially by agents who are new users of the system and have not yet established their own trust relations that can be the basis of transitive propagation (solving a kind of "cold start problem"). Notice here that the notion of similarity propagation can be easily extended to the use of *stereotypes* by humans. A stereotype may be thought of as an artificial construct that possesses certain characteristics similar to other humans. It can be modeled as an artificial node in the trust (distrust) relation that is not associated with any real agent. However, the similarity relations used in both trust and distrust propagation may be applied to real agents and to the stereotype. In this way, trust and distrust may propagate based on similarity to a stereotype; this kind of propagation is especially useful if the agent has no other information available (for example, no basis of comparison of the newly encountered agents to any previously encountered agents). Of course, similarity propagation based on stereotypes may lead to wrong results, but this objection applies to all kinds of human trust propagation. The possibility of making a wrong judgement of trust

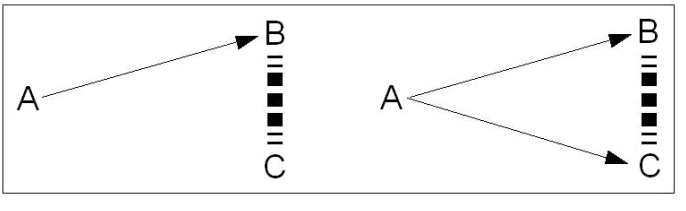


Fig. 2.5 Combination of Similarity and Transitive Propagation of Trust

(distrust) towards a stranger can never be excluded, and in such a case the trust (distrust) relation will be corrected using first-hand experience (which is a different phenomenon than trust propagation).

This kind of propagation is related to observations that trust is correlated with acquaintance and social similarity. This observation has been supported by numerous sociological and psychological studies [119, 156, 103]. On the other hand, it should be clear that similarity propagation needs to be well defined and studied before it is applied in practice. Not all similarity relations can be used for similarity propagation. Not only does the similarity relation need to be precisely defined, but it may also be necessary to define a degree of similarity and specify how strong the trust relation will be that is the result of similarity propagation given the degrees of similarity between agents.

Another important aspect of propagation is the consideration of distrust. It is clear that transitive propagation does not apply to distrust, because if A distrusts B and B distrusts C, it does not follow that A will distrust C – perhaps on the contrary (but not necessarily). On the other hand, similarity propagation may apply to both trust and to distrust, although the meaning of the similarity relations should be different in both cases. On the other hand, the propagation of trust and distrust jointly can be transitive, even if only in a limited manner. If A trusts B, B distrusts C and gives a negative (distrustful) recommendation about C to A, then A may distrust C. This means that trust can be propagated transitively with distrust, but not the other way around.

The kinds of human trust propagation described in this section have also been used to support computational trust. Efforts to validate computational trust have also partially validated these kinds of trust propagation [57]. For more on the subject of computational trust propagation and the validation of computational trust, see chapter 3.6.3.

The subject of human trust propagation should become one of the central issues of trust research, as it is these kinds of propagation that really extend the trust network to strangers. Human trust propagation is important, but it should be noted that it is very difficult to replicate human reasoning about trust using computational trust and simple computational trust propagation rules. On the other hand, using properties of human trust propagation, it would be possible to formulate na *operational definition of computational trust* that would not have to rely an definitions of trust formulated previously. Any relation between agents that would satisfy the properties of human trust propagation could be referred to as trust, using the operational definition.

2.2.8 Probabilistic Trust Estimation

The trust definitions discussed above do not preclude a probabilistic interpretation of trust. In particular, it is possible to define trust as a probability distribution of agent behavior. Then, expectancy and dependency trust can be viewed merely as the two moments of a probability distribution.

This point of view is attractive because it immediately gives a set of mathematical tools for the estimation of trust. However, a number of assumptions need to be made in order to apply these tools. The following assumptions are required:

- agents behave in a probabilistic manner (each agent has an innate probability for each possible kind of behavior),
- behavior of agents is independent,
- an a priori distribution form of agent behavior is known.

If the above assumptions can be made (to an extent, this is possible for example for artificial agents), then trust estimation might proceed in the following manner. For simplicity, let us assume that agents can choose only two actions in an encounter that represent fair and unfair behavior. These actions are chosen by each agent B with a probability p_B. Assume that agents share reports about their behavior with each other, but can also falsify these reports. Let us denote the probability of lying in a report by agent C as l_C (note that this is in fact a reputation in the context of credibility). The probability of receiving a report rep_C about agent B from agent C is as follows:

$$P[Rep_C = rep_C] = \begin{cases} l_C(1-p_B) + (1-l_C)p_B \text{ if } rep_C = 1 \\ l_C p_B + (1-l_C)(1-p_B) \text{ if } rep_C = 0. \end{cases}$$

When an agent A searches for information about an unknown agent B, the probability of receiving a set of reports $\{rep_{C_i}\}$ from various agents C_i is a function of the probability that agent B behaves unfairly (given that the credibility probabilities are known and independent of p_B). The probability of receiving a random set of reports is given by:

$$P[\{rep_{C_i}\}] = P[Rep_{C_1} = rep_{C_1}]P[Rep_{C_2} = rep_{C_2}] \ldots \\ \ldots P[Rep_{C_n} = rep_{C_n}].$$

A maximum likelihood estimation of p_B can be applied when agent A knows the values of received reports. This estimation of p_B relies on the fact that the probabilities of lying in reports (credibility reputation) l_C is known. In some applications, these probabilities can be determined through a comparison of agents A's own observation with the reports received from other agents (this is possible for example in recommender systems). Note that such a use of credibility reputation is in fact an application of the transitive propagation of trust in different contexts (because l_C is reputation credibility and p_B determines trust in a different context).

The mathematical approach presented here for the simple case of encounters with just two actions can be extended to more complex scenarios. However, the fundamental assumptions presented above need to hold in order to apply this method. In more complex scenarios, a distribution form must be

assumed for the distribution of agents' actions. [118] has assumed that the proportion of the number of received positive reports to all reports (still for encounters with two actions) has a Beta-prior distribution. The parameters of this distribution can be adjusted to suit various a priori assumptions about agent behavior, for example an assumption of ignorance that requires a uniform distribution.

In the example shown above, transitive propagation can be applied for just one step (probabilistic methods do not support similarity propagation). One of the inherent difficulties in applying the probabilistic method is that in order to apply transitive propagation for more than one step using Bayesian estimation, it is necessary to deal with multiple propagation paths (and paths that can include cycles). Applying Bayesian estimation requires a selection of single paths without cycles. In section 3.6.3, different propagation algorithms that are capable of selecting such paths will be described.

2.2.9 Other Concepts of Trust Theory

2.2.9.1 Confidence

The distinction between confidence and trust is mostly due to Luhman [101]. Luhman defines *confidence* as *an expectation that neglects a small possibility of disappointment in a situation where there is no alternative but to expect a positive outcome*. For example, confidence may be placed in the fact that our car will not suddenly break down on the road, or that a war will not suddenly be started by our government (this example shows that confidences, too, can be misplaced). The main difference between confidence and trust is that when humans use trust, they must have alternatives available. In other words, we can say that trust is confidence, if there is no alternative but to make a trusting choice.

2.2.9.2 Self-trust

The notion of self-trust has been introduced in [31] as a basis for all human trust relationship. Self-trust is a precondition for the establishment of any trust relation. Considering transitive trust propagation discussed in the previous section, it can be seen that if an agent does not trust himself, it would follow that this agent cannot trust any other agent. In fact, this view has been confirmed by sociological research that has linked the lack or trust (or the existence of distrust) to low levels of social intelligence and self-esteem [188, 117].

2.2.10 Other Trust Definitions

Research on trust is abundant in the social sciences. [171] gives a very good review of research in trust in sociology, psychology and anthropology. The theoretical overview presented in this chapter focuses on trust definitions and their implications for computational trust expression.

The definitions discussed above are based on previous research, but have also been adapted and sometimes modified in an attempt to create a set of logically consistent theoretical terms and relations. In this attempt, some of the finer points of the definitions of human trust available in literature may have been lost. In this section, we summarize the most important research in the area, quoting verbatim the most important definitions from the considered texts. At the same time, quoted definitions will be compared with the definitions previously introduced in this chapter.

2.2.10.1 Luhman

For Luhman [101], trust is:

> an effective form of complexity reduction.

To understand this definition, it is necessary to comprehend Luhman's notion of complexity:

> complexity is manifested as the unimaginable superabundance of the world's realities and possibilities.

A human's social environment further increases the world's complexity. Therefore, Luhman observes:

> in conditions of increasing social complexity, man can and must develop more effective ways of reducing complexity.

Luhman's trust definition seems, at first, completely different from the discussion in this chapter. The main reason for this apparent difference is that Luhman's definition is not psychological, but sociological; it does not attempt to describe the mental state of humans that is trust. Rather, Luhman attempts to describe how trust operates in a society. However, it seems that Luhman has also unwittingly described a psychological notion. Trust can be seen as a mental heuristic that is used by humans in order to reduce the high complexity of reality. Instead of constantly using complex deliberative reasoning humans apply such heuristic in order to make decisions faster, or when they do not have sufficient information. The origin of such heuristics is largely evolutionary. For this reason, Luhman's definition does not contradict our notions of expectancy and dependency trust; rather, it shows a different aspect of trust, and stresses the value of trust in human decision making in a social environment.

2.2.10.2 Gambetta

Gambetta's [53] definition of trust is as follows:

> trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action.

Gambetta's definition is consistent with expectancy trust. While the definition uses the term of "subjective probability", this can be extended to an expectation, as has been explained whilst discussing the relationship between expectancy trust and reputation.

2.2.10.3 Mui

Mui's [118] definition of trust is one of the most simplest ones:

> Trust: a subjective expectation an agent has about another's future behavior.

The definition of Mui is based on Gambetta's, and is very similar to our definition of expectancy trust. Mui's research has given much support to the hypothesis that trust and reputation are necessary for the evolution of reciprocity and altruism [115].

2.2.10.4 Deutsch

The trust definition of Deutsch [36] states:

> 1. The individual is confronted by an ambiguous path, a path that can lead to an event perceived to be beneficial (Va^+) or to an event perceived to be harmful (Va^-);
> 2. he perceives that the occurrence of Va^+ or Va^- is contingent on the behavior of another person;
> 3. he perceived the strength of Va^- to be greater than the strength of Va^+.
>
> If he chooses to take an ambiguous path with such properties, I shall say that he makes a trusting choice; if he chooses not to take the path, he makes a distrustful choice.

The definition of Deutsch is quite complex, and is further refined in his work. However, it is possible to relate the quoted definition to the terms used in this book. Firstly, Deutsch presupposes a situation of uncertainty or risk that is due to the behavior of independent persons, as in our notion of an encounter between agents in the ODS. Secondly, Deutsch assumes that trust is "perceived", is similar to considering trust as subjective. Thirdly, Deutsch introduces a condition that is not used in our definitions, that the harm must

outweigh the benefit of the situation. This condition is rather strong, but could be reformulated as the assumption that a risk must exist in the encounter. Clearly, if all outcomes of actions in an encounter will give an equally good outcome, trust is not required. Therefore, the trust definition of Deutsch is partially consistent (although stronger) with our definition of dependency trust.

2.2.10.5 Marsh

In [107], Marsh defines trust as:

> the belief (or a measure of it) that a person (the trustee) will act in the best interests of another (the trustor) in a given situation, even when controls are unavailable and it may not be in the trustee's best interests to do so.

In his thesis, March deals mostly with computational trust [108]. He introduces a formalism that aims to replicate how human trust works. The formalism is largely based on the work of Deutsch on human trust, as it aims to reproduce Deutsch's psychological observations of how human trust is used.

2.2.10.6 Barber

Barber's [9] view of trust is inherently sociological. He recognizes three dimensions of trust that can be an expectation of:

- the persistence and fulfillment of the natural and moral social orders,
- technically competent role performance,
- that partners in interactions will carry out their fiduciary[4] obligations and responsibilities, that is, their duties in certain situation to place others' interests before their own.

Barber does not give a precise definition of trust, but rather stresses some sociological aspects of trust. His conclusion is very similar to the work of Elgesem, and therefore will be discussed in more detail below.

2.2.10.7 Elgesem: Normative Trust

In [44], Elgesem gives a definition of trust that is in many ways consistent with Barber's view:

> trust is a normative notion in the sense that an essential ingredient in all cases of trust and trustworthiness is the existence of a set of norms that provide the motivation to cooperate.

[4] The fiduciary duty is a legal relationship of confidence or trust between two or more parties, most commonly a fiduciary or trustee and a principal or beneficiary. Source: Wikipedia, the free encyclopedia.

The definitions of Elgesem is consistent with our refined definitions of expectancy trust. Recall from the discussion of positive and negative expectancy trust that trust could be an expectation about the behavior of a trustee based on a model of his reasoning. Elgesem's and Barber's views on trust support this more general definition: in their case, the expectation concerns that the trustee applies certain norms of behavior. Barber states that these norms can be the "natural and moral social order", the norms that describe technical competence and professionalism, or the norms that govern legal obligations. t Elgesem gives a more abstract view of "norms that provide the motivation to cooperate". In [178], trust is redefined as an expectation of or dependency on fair behavior. Using the definition of fairness as a justified expectation, we have shown in the discussion above that the concepts of expectancy trust and fairness can be logically related: using these concepts, the statement that "a trustor can trust the trustee if the trustee behaves fairly" is a conclusion from the two definitions. These considerations all point to the relation between trust and fairness that will be the subject of much of this book.

2.2.11 Economic Theories of Trust

Most of the trust definitions described in this chapter have been applied in economics, sociology, psychology, anthropology and many other social sciences. Quite apart from theories of human trust that aim to explain how humans use trust in situations of uncertainty, economic theories of trust deal with the subject of how trust impacts entire economies, but also how trust impacts prices. This research has largely avoided the discussion of human trust, sometimes by using reputation and investigating its influence on behavior directly (without considering trust), and sometimes by simply adopting a behavioral model of human trust (for example, seeing trust as a state of mind that allows users to cooperate). In this section, we shall briefly summarize the findings of economic theories of trust in order to understand how trust can impact large-scale ODS (of which an economy is a good example). Another reason for a discussion of this research is that it has a significant empirical background.

In economics, trust is often defined as a factor that decreases transaction costs. [192] develop a theoretical model where trust is defined as the time people spend in production rather than in verifying that others do not cheat or behave opportunistically. This definition of trust is consistent with dependency trust, as high dependency trust leads agents to tolerate risks. This means that in an encounter where one action implies a dependence on the trustee, while another allows a reduction in risk through costly verification, the trustor will choose the first (more risky, but cheaper) action if he has sufficiently high dependency trust. In the well known research of [136], one finding is that the more an individual trusts others, the less costly his/her daily social interactions will tend to be. This view includes economic interactions, and once again is consistent with the definition of dependency trust.

Neuroscience also provides empirical support for research on trust. In [78], authors found that reciprocity expressed by one player strongly predicts future trust expressed by their partner, measured by neural responses in the dorsal stratum of the brain.

2.3 Theory of Distributive Fairness

The general definition of fairness used in this book is that of a "justified expectation". The definition goes on to say that the expectation can be justified "according to rules that apply in a specific context based on reason or precedent". This section gives an overview of the various "rules based on reason" that can be applied to a distribution problem. While this section focuses solely on fairness in distribution problems, let us not forget that the "justified expectation" used in the definition of fairness may also be the result of an agreement that does not concern a distribution of goods, resources or costs – for example, the agreement on following the rules of a computer game.

The problem of distributing resources or costs in an ODS may be simple if the rules of fairness are indeed established by precedent. This is sometimes the case, when the ODS applies rules that are already established in real society. Such is, for example, the case of e-commerce, where fairness is simply defined as satisfaction of justified expectation according to rules that are based on commercial law. Yet, even here, the specific circumstances of the ODS may lead to problems of fairness that are unexpected in the real world. Note that we want to exclude from consideration adversary behaviors in trust management systems, although these, also, could be called unfair. However, in online auctions, there exist opportunities for unfair behavior in auctions that are much less common in the real than in the virtual market. Also, in some markets a regulator is required, and then the question of fair distribution by the regulator remains an issue.

However, rules that are based on precedent, norms or codes of behavior may not always suffice for an expression of fairness in the ODS. When they do not, the definition allows the use of rules based on reasoning. This means asking the question: what rules of distribution, or what solutions of the distribution problem, would be fair in the given context? The answer is not always obvious. Consider the problem of several agents using a computer network that need to share a common bandwidth for different purposes, and have different throughput requirements. In addition, the operator of the network is also a party of the distribution problem, and is the interested primarily in most cost-effective allocation of his resources. The search for a fair throughput distribution in such a situation is far from simple. Still, it can be approached as a mathematical problem.

Contrary to trust and trust management, distributive fairness has received much attention from mathematicians. There exist several mathematical (also axiomatic) formulations of distributive fairness, as well as entire theories of fairness [95, 47, 149]. In this section, we shall also formulate a theory of

distributive fairness that uses a multi-criteria optimization model of the fair distribution problem. This theory, referred to in this book as the *theory of equitable optimality*, will be presented in the section 2.3.2.

Several algorithms and mechanisms can be applied in various contexts of fair distribution problems. For example, for the purpose of distributing indivisible goods or costs, a priority queue can be used. An example in the ODS would be the selection of a peer to serve as a superpeer in an overlay P2P network. The superpeer must provide some of his resources to other peers, yet the network as a whole often works more effectively when superpeers are used. Who should become the superpeer, and under what conditions? Making all peers become superpeers in turn is not a good solution, as some peers do not have sufficient resources. While a priority queue can be used here, it is still not obvious what should determine the priority, and priorities will probably depend on multiple factors. The algorithms and mechanisms to solve fair distribution problems in an ODS will be discussed in more detail in chapter 4.

In many of the distribution problems in the ODS, game-theoretic formulations can be used. An example could be that of a grid system that is shared by several agents; each of these agents is selfish and attempts to optimize his job execution times. Clearly, such a distribution problem cannot always be solved; an agent that has few computational jobs may execute them all locally and will get best performance if he does not allow any other agents to use his resources. This example shows that agents can withdraw their resources if they believe that they are not treated fairly, or if they are selfish and believe that participating in the ODS does not lead to an advantage. Since the grid depends on agents who contribute resources, this is not the best solution. It would be better to provide incentives for the agent to participate and provide resources for others, receiving resources at a later time when he needs them. Still, the question of how much resources an agent should supply is a distribution problem that should be solved fairly, considering the interests of all involved agents. Game-theoretic formulations of fair distribution problems will be considered in section 2.3.5.

2.3.1 Comparing Agents to Establish Fairness

Before going on to discuss the various concepts and theories of distributive fairness, it is necessary to mention a basic problem. In order to make any decision about the fairness of distributions, or to execute a procedure that aims to establish a fair distribution, it is necessary to make comparisons between different agents. This necessity is obvious, but raises methodological questions when using the concept of utilities. The theory of utility usually disallows comparisons between utilities of various agents, deeming such utilities incomparable. Also, in a procedure it would not be safe to assume that an agent knows the utility of another agent (game theory predicts that depending on the availability of this knowledge, outcomes of games could differ).

For this reason we shall initially assume comparisons only between the directly measurable or observable outcomes of agents. In other words, whenever a distribution problem is considered, there must be some measurable criteria for evaluating the outcome of any agent (in units of money, resources, goods, or any other). Comparisons between agents' outcomes are made possible by using these criteria. The theory of equitable optimality can thus be formulated without using the concept of agent utility. Whenever this concept is required for some theoretical consideration (as in game theory), this assumption will be stated explicitly. This is the case with economic theories of fair allocations, described in section 2.3.8, that are based on the concept of no-envy. In section 2.3.4, this assumption is relaxed.

Note that we assume that all agents are equally entitled or capable of achieving good outcomes. We shall call such agents *similarindexsimilar agents agents*. The theory of equitable optimality can be extended to take into account various priorities of agents, but this makes the definition considerably more complex [122]. This extension of the theory will be discussed in section 2.3.3. If the agents are not similar because they have different levels of expenditure or contribution and are therefore entitled to different outcomes, a common practice is to transform every agent's outcome by dividing them by the agent's contribution [173]. After such a transformation, it is possible to think of the agents as similar, because they are equally entitled to receive a unit of outcome per unit of contribution. If some agents are not similar for other reasons (in an Internet auction, the reason can be that various sellers have various qualities of goods or services, and various marketing), then it is still possible to consider the fairness for a subset of agents that are similar according to these criteria. A system should at least be able to provide fairness to this subset of similar agents. This approach is equivalent to a ceteris paribus assumption from economics: *when all other factors can be excluded* and all agents are equally entitled, the theory of equitable optimality can be used for testing distributional fairness. In a laboratory setting, such conditions can often be satisfied and we can design systems that realize the goal of fairness, even in the presence of adversaries that do not act in a procedurally fair manner.

2.3.2 Equitable Optimality

In a fair distribution problem, the objectives of all agents must be taken into consideration. The problem of optimizing the outcomes of all agents can be formulated as a multicriteria problem. We shall refer to this as the problem of *efficient optimization*. Efficient optimization of an agent's outcomes need not have any concern for fairness. The outcomes can be the shares of goods or costs received by agents in an ODS. Let $\mathbf{y} = [y_1, ..., y_n] \in Y$ be an outcome vector of the efficient optimization problem where Y is a set of all possible outcomes. We assume that there are n agents that maximize their outcomes without loss of generality: if outcomes are minimized, it is enough to multiply

them by -1 in order to transform the efficient optimization problem into a maximization problem.

Because of our assumption that there exists a common, objective, measurable criterion for evaluating the outcome of any agent, we can assume that the values of y_i are all specified in the units of that criterion. For simplicity, let us assume that y_i are non-negative real numbers.

Outcomes of the efficient optimization problem can be for example the download speeds of agents in a Peer-to-Peer file sharing system. Another example is the amount of computational time received in a grid. In grids, it is also possible to use outcomes that should be minimized, for example the time needed to compute all the jobs of an agent. For TCP flows that share an IP network, the outcomes may be the throughputs of each flow; the values of these outcomes depend on the routing in the IP network that can use traffic engineering in order to increase fairness among the flows. Note that in the previous example, network management may also be interested in throughput maximization, rather than fairness. Other examples can include the problem of locating a facility in a city (for example, a hospital or a school). The travel distances of the users of this facility (for example, hospital patients) can be the outcomes of such an efficient optimization problem.

Among general examples of the fairfair distribution problem, the famous cake-cutting problem should be mentioned. The outcomes of this problem are the shares of agents participating in the division of cake that has a fixed size. The fair solution of this problem is known (if agents are similar, they should have equal shares), but the problem often focuses on fair procedures for the division of cake, which makes this a problem of procedural fairness. A more interesting, general example is a fair distribution problem with a budget. In this problem, each agent is associated with a cost, and the sum of outcomes of all agents multiplied by their costs cannot exceed the available budget.

In order to make this approach to solving distribution problems operational, one needs to assume some solution concept specifying what it means to maximize multiple outcomes. The solution concepts are defined by properties of the corresponding preference model within the outcome space.

The commonly used concept of the (strongly) Pareto-optimal solutions, as feasible solutions for which one cannot improve any criterion without worsening another, depends on the dominance relation $\succeq_r$ which may be expressed in terms of the vector inequality: $\mathbf{y}' \succeq_r \mathbf{y}''$ iff $y'_i \geq y''_i$ for all $1 \leq i \leq n$, where at least one strict inequality holds.

Without any additional constraints, any Pareto-optimal solution of this problem will be an *efficient solution*. In plain words, this means that anything goes – a solution where one agent gets all, and nobody else gets anything, is as good as a solution where all agents get equal shares.

Distributive fairness can be based on an axiomatic expression and can be expressed as another multicriteria optimization problem using the theory of equitable optimality (this definition is also referred to as equitable optimization [128]). The axioms of the theory of equitable optimality define a

preference relation on the outcome vectors of the efficient optimization problem. An equitable rational preference relation is any *symmetric and transitive* relation satisfying the following axioms [128]:

1. **Impartiality.** The ordering of the outcome values is ignored (e.g. a solution $y = [4, 2, 0]$ is equally as good as a solution $y = [0, 2, 4]$). First of all, fairness requires impartiality of evaluation, thus focusing on the distribution of outcome values while ignoring their ordering. This means that in the efficient optimization problem we are interested in a set of outcome values without taking into account which outcome takes a specific value. Hence, we assume that the preference model is impartial (anonymous, symmetric). In terms of the preference relation it may be written as the following axiom. Let $I = \{1, 2, \ldots, n\}$

$$(y_{\tau(1)}, y_{\tau(2)}, \ldots, y_{\tau(n)}) \cong (y_1, y_2, \ldots, y_n) \quad \text{for any permutation } \tau \text{ of } I, \tag{2.1}$$

   which means that any permuted outcome vector is indifferent in terms of the preference relation.
2. **Monotony.** An outcome vector that improves the value of one of the outcomes is preferred, the values of other outcomes are not deteriorated (e.g. $y = [4, 2, 0]$ is preferred to $y = [3, 2, 0]$). This axiom is actually a repetition of the requirement of efficiency. It guarantees that only efficient solutions will be chosen as equitable solutions. Another way of looking at it is that the monotony axiom prevents a phenomenon well-known in former Communist countries: that of "equating downwards", or making outcomes worse (and more equal, but not more equitable) for everyone. The axiom can be expressed as follows:

$$\mathbf{y} - \varepsilon \mathbf{e}_i \prec \mathbf{y} \quad \text{for } \varepsilon > 0,\ 1 \leq i \leq n. \tag{2.2}$$

3. **Principle of transfers.** A transfer of any small amount from an outcome to any other relatively worse-off outcome results in a more preferred outcome vector (e.g. $y = [3, 2, 1]$ is preferred to $y = [4, 2, 0]$). Fairness requires equitability of outcomes which causes that the preference model should satisfy the (Pigou–Dalton) principle of transfers. The principle of transfers states that a transfer of any small amount from an outcome to any other relatively worse–off outcome results in a more preferred outcome vector. This axiom may be expressed as a property of the preference relation:

$$y_{i'} > y_{i''} \quad \Rightarrow \quad \mathbf{y} - \varepsilon \mathbf{e}_{i'} + \varepsilon \mathbf{e}_{i''} \succ \mathbf{y} \quad \text{for } 0 < \varepsilon < y_{i'} - y_{i''}. \tag{2.3}$$

The above axioms are sufficient in order to compare two outcome vectors on the basis of theory of equitable optimality, to determine whether they are equitably equivalent, or whether one of them is preferred to another. For instance, consider the two vectors: $y_1 = [5, 5, 10]$ and $y_2 = [2, 2, 2]$. It can be seen that by applying the axiom of monotony, vector y_1 is equitably preferred

to y_2. This example demonstrates that the *theory of equitable optimality is not about equality*: vector y_2 has an equal distribution of outcomes, but it is considered worse, because in outcome y_1, all agents are better off. On the other hand, the two vectors: $[2, 4, 8]$ and $[2, 5, 6]$ are equitably incomparable: none of them can be preferred to the other. *Equitable solutions* of the efficient optimization problem are the solutions that cannot be improved in the preference relation established by the axioms of the theory of equitable optimality.

Any relation on the outcome vectors of the distribution problem that is symmetric, transitive and satisfies axioms (2.1), (2.2) and (2.3) will be called hereafter a *fair (equitable) rational preference relation*.

We say that outcome vector $\mathbf{y}'$ *fairly dominates* $\mathbf{y}''$ ($\mathbf{y}' \succ_e \mathbf{y}''$), iff $\mathbf{y}' \succ \mathbf{y}''$ for all fair rational preference relations $\succeq$. In other words, $\mathbf{y}'$ fairly dominates $\mathbf{y}''$, if there exists a finite sequence of vectors $\mathbf{y}^j$ $(j = 1, 2, \ldots, s)$ such that $\mathbf{y}^1 = \mathbf{y}''$, $\mathbf{y}^s = \mathbf{y}'$ and $\mathbf{y}^j$ is constructed from $\mathbf{y}^{j-1}$ by application of either permutation of coordinates, equitable transfer, or increase of a coordinate.

To understand the fair preference relation better, it is good to start by considering just two of the axioms: impartiality and monotony. Together, these two axioms define a set of preference relations that is larger than the set of fair preference relations. Let us limit our attention to an unconstrained distribution problem with just two agents (where each outcome vector $\mathbf{y} = (y_1, y_2)$, y_1 and y_2 are positive real numbers). In such a problem, shown in Figure 2.6, preference relations that conform to the axioms of monotony and impartiality allow us to distinguish between the set $D(\mathbf{y})$ of dominated outcomes (obviously worse than $\mathbf{y}$ for all fair preferences) and the set $S(\mathbf{y})$ of dominating outcomes (obviously better for all fair preferences). To see why the solution $\mathbf{y}'$ shown in the figure is better than $\mathbf{y}$, consider that according to the axiom

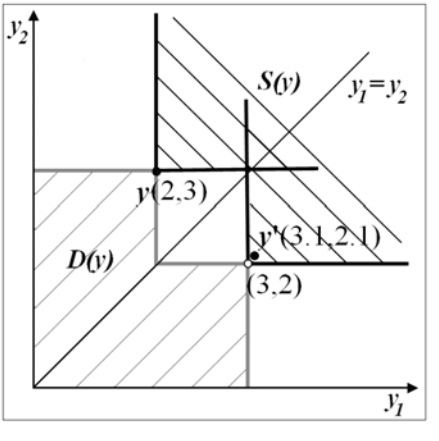


Fig. 2.6 Ordered Pareto preference relation in 2D: $D(y)$ - set fairly dominated by y, $S(y)$ - set of outcomes fairly dominating y

of impartiality, the outcome $(3, 2)$ is equally preferable as $\mathbf{y}$; and according to the axiom of monotony, $\mathbf{y}' = (3.1, 2.1)$ is better than $(3, 2)$. Since a preference relation is also transitive, $\mathbf{y}'$ is preferred to $\mathbf{y}$. Another way of looking at the sets D and S is as follows: notice that as we order outcome $(3, 2)$ starting from the worst outcome to the best, we get $(2, 3)$. And, the set $S(\mathbf{y})$ is composed of the solutions that Pareto-dominate $\mathbf{y}$. Similarly, the set $D(\mathbf{y})$ is composed of the solutions that are strongly Pareto-dominated by $\mathbf{y}$. Thus, the axioms of monotony and impartiality define preference relations that correspond to the ordered strong Pareto dominance over Y (where the ordering is from worst outcome to best). This set of solutions plays an important role in the theories of fair distribution. The theory of equitable optimality defines a subset of this set (this subject is discussed in further detail in section 2.3.4).

Equitable solutions are visualized in Figure 2.7. Notice that the set $S(\mathbf{y})$ changes from Figure 2.6 because of the addition of the third axiom: the principle of transfers. This axiom implies that all solutions that have a sum of outcomes similar to $\mathbf{y}$, but where both outcomes are more equal, are preferred to $\mathbf{y}$. These solutions lie on the black line between the two points shown in the figure. The axiom of monotony implies that all solutions that have a larger coordinate than any of the solutions on this line are also preferred. This increases the set $S(\mathbf{y})$ when compared to figure 2.6. Similarly, the set $D(\mathbf{y})$ is increased, because solutions that have a similar sum of outcomes as y, but are more unequal, are considered as worse than $\mathbf{y}$. However, some outcome vectors are left (in white areas) and they can be differently classified (as dominated or not) by various specific fair preferences.

Another graphical description of the theory of equitable optimality is shown on figure 2.8. The left part of the figure shows one of the key concepts of distributive fairness: the Generalized Lorenz curve (to use its name from

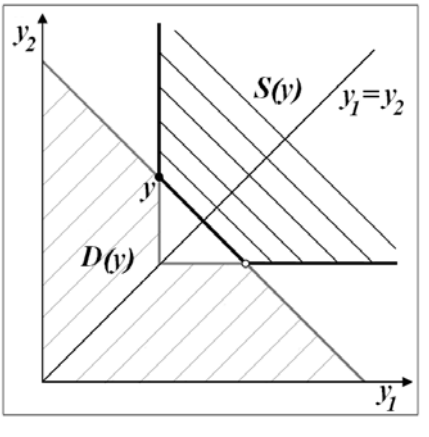


Fig. 2.7 fair preference relation in 2D: $D(y)$ - set fairly dominated by y, $S(y)$ - set of outcomes fairly dominating y

economics). The Generalized Lorenz curve is obtained by first taking the outcomes of all agents that participate in a distribution and ordering them from worst to best. Then, the ordered outcomes are added one at a time, creating a sequence of cumulative sums, called *cumulative ordered sums*. Usually, the Generalized Lorenz curve is normalized by the number of agents, n [150]. In this book, we use a re-scaled version that is not divided by n, which is useful because the curve can then be used to show the sum of total outcomes of all agents.

Let us denote this operation by a vector function $\theta(\mathbf{y}) = [\theta_1(\mathbf{y}), ..., \theta_n(\mathbf{y})]$ of the outcome vector. The cumulative ordered sums of agents' outcomes are calculated starting from the outcome of the worst agent (θ_1), then the sum of outcomes of the worst and the second worst (θ_2), and so on, until the sum of all agent outcomes, which is denoted in the figure as θ_n. The second line in the figure, the equal distribution line, is simply a straight line connecting the points $(1, \theta_1)$ and (n, θ_n). The area between the two curves, denoted by S, can be seen as a measure of inequality of the agent's outcomes. The objective of distributive fairness is to minimize this inequality, making the Generalized Lorenz curve as close to the equal distribution line as possible. Note that this objective frequently forms a tradeoff with the objective of maximizing the total sum of agents' outcomes (θ_n). The right part of Figure 2.8 shows two generalized Lorenz curves that correspond to different distributions among the same agents. The first distribution has a higher θ_n, but also a higher inequality, while the second distribution has a lower total of agents' outcomes, but is fairer. In terms of equitable optimization, the two distributions on the right side of Figure 2.8 are incomparable - the choice of one of them depends on the preferences of a decision maker[5].

By applying cumulative ordered sums, the efficient optimization problem is transformed into another multicriteria optimization problem. Therefore,

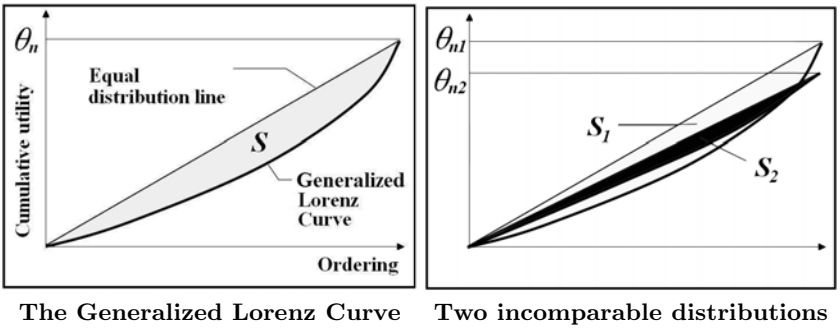


Fig. 2.8 Examples of Generalized Lorenz curves

[5] Incomparable distribution can also have identical total efficiencies θ_n.

the Generalized Lorenz curve actually represents the criteria that need to be optimized in order to achieve equitably efficient (fair) solutions. These criteria are the cumulative sums of worst outcomes $\theta_1, \theta_2, ..., \theta_k$. The new optimization problem will be called the *equitable optimization* problem. The Pareto-optimal solutions of the equitable optimization problem are a subset of the efficient solutions. More importantly, they are equitable solutions of the efficient optimization problem: they cannot be improved in the fair preference relation. This observation is established by the following theorem (a consequence of results from the theory of majorization):

Theorem 1. *[128] Outcome vector* $\mathbf{y}' \in Y$ *fairly dominates* $\mathbf{y}'' \in Y$*, iff* $\theta_i(\mathbf{y}') \geq \theta_i(\mathbf{y}'')$ *for all* i *where at least one strict inequality holds.*

Note that Theorem 1 permits us to express equitable efficiency for the efficient optimization problem in terms of the strong Pareto-optimality for the equitable optimization problem:

Corollary 1. *A feasible solution* $y \in Y$ *is an equitable solution of the efficient optimization problem, if it is a Pareto-optimal solution of the equitable optimization problem, the problem with objectives* $\Theta(\mathbf{y})$*:* $\max \{(\theta_1(\mathbf{y}), \ldots, \theta_n(\mathbf{y}))\}$.

Note that according to the theory of equitable optimality, the utilitarian approach is a special case: optimizing the sum of an agent's utilities also leads to an equitably efficient solution [128]. However, this solution is one of many and it is up to the decision maker (the designer of a fairness management system) to make a choice for one solution. At the other extreme, the solution obtained by lexicographically optimizing the outcomes of all agents (without constraints, this approach would equalize outcomes) is also equitably efficient. The lexicographic optimization approach starts with optimizing the outcome of the worst-off agent, then the second-worst off, and so on, until a constraint is reached or the ordering of the agents changes – the approach continues until all agents have active constraints, or the outcomes are equal. In chapter 4, we shall discuss various methods of solving equitable optimization problems. Many of these methods allow for the expression of various preferences that lead to different equitable solutions.

The area between the Generalized Lorenz curve and the equal distribution line can be simply calculated and used as a computable measure of inequality. It can be shown that minimizing this measure leads to fair distributions [128]. The Gini coefficient (frequently used in economics) is the area S normalized by θ_n: $Gini = \frac{2S}{\theta_n}$. Note that minimizing the Gini coefficient to obtain fair distributions can lead to worse total outcomes (sums of all agent's utilities). Also, the Gini coefficient is not consistent with fair preference: a distribution that has a smaller Gini coefficient can be dominated in terms of the fair preference relation by another distribution. This fact is a consequence of normalization.

Another measure of fairness that is consistent with equitable optimality is the area under the Generalized Lorenz curve (BLC), $BLC = \frac{n\theta_n}{2} - S$. This area is always larger if the distribution D_1 is equitably preferred to another distribution D_2, since the Generalized Lorenz curve of D_1 is then above the Generalized Lorenz curve of D_2. Unfortunately, maximizing the area under the Generalized Lorenz curve does not necessarily always lead to a more equitable solution. As in the example shown in 2.8, the areas under two incomparable generalized Lorenz curves may differ, but if one is larger this does not imply that the solution is more equitable. A full solution of the equitable optimization problem can only be achieved by finding a set of Pareto-optimal solutions and choosing one that best suits the preferences of a decision maker (which are expressed as additional input to the problem). The decision maker can then choose whether or not he prefers increase equality at the cost of decreasing the total outcome.

2.3.3 Considering Entitlements in Equitable Distribution

Some of the most popular definitions of equity [39] consider that equity is always related to the concept of agent entitlements. In our previous discussion of the theory of equitable optimality, we assumed that all agents are equally entitled. This section will demonstrate why this is not a significant simplification of the theory and how entitlements can be introduced into equitable optimization.

Agent entitlements are usually modeled as weights that indicate an agent's priority or importance. These weights can be determined from an agents', level of past contribution or cost (for example, based on the amount of provided resources in a P2P or grid system). For simplicity, we shall assume that the agents in the ODS have weights v_i that are positive integer numbers. Sometimes we will use normalized weights $\overline{v}_i = v_i / \sum_i v_i$.

The original problem of efficient optimization is now redefined: there are n agents in the ODS with outcomes y_i and weights v_i. The question is: what is a good definition of equitable solutions for such a problem?

The efficient optimization problem with weights can be transformed into an unweighted problem (in which all agents have weights equal to 1). For integer weights, this can be simply done by splitting each weighted agent i into v_i unweighted agents. The outcomes of the split agents can be added in order to determine the outcome of the original, weighted agent. Thus, the equitable solution for the weighted problem is defined as the equitable solution of the transformed unweighted problem.

Note that this transformation can also be applied for rational weights, but usually dramatically increases the problem size. For this reason, it is also useful to consider methods of finding equitable solutions for the weighted problem directly. Such methods will be considered in section 2.3.9.

The Generalized Lorenz Curve used to represent the theory of equitable optimality can be reformulated for the weighted problem. This reformulation uses the cumulative distribution function of the distribution of weights according to sorted outcomes:

$$F_y(d) = \sum_i \overline{v}_i \delta_i(d), \;\; \delta_i(d) = \begin{cases} 1 & \text{if } y_i \leq d \\ 0 & \text{otherwise.} \end{cases}$$

The integral of the inverse of this cumulative distribution function is the Generalized Lorenz Curve for the weighted problem. For more details, the reader is referred to [122].

It is also possible to redefine the axioms of the theory of equitable optimality so that they can be applied directly to the outcomes of the weighted problem. The axiom of monotony does not need to change. The axiom of impartiality can be replaced by the equality of distributions defined above:

$$F_{y'} = F_{y''} \quad \Rightarrow \quad y' \cong y''. \tag{2.4}$$

The principle of transfers can be redefined using normalized weights so that the transfer is done in weighted units.

$$\frac{y_{i'}}{\overline{v}_{i'}} > \frac{y_{i''}}{\overline{v}_{i''}} \quad \Rightarrow \quad \mathbf{y} - \varepsilon \overline{v}_{i'} \mathbf{e}_{i'} + \varepsilon \overline{v}_{i''} \mathbf{e}_{i''} \succ \mathbf{y} \quad \text{for } 0 < \varepsilon < \frac{y_{i'}}{\overline{v}_{i'}} - \frac{y_{i''}}{\overline{v}_{i''}}. \tag{2.5}$$

2.3.4 A Utility-Based Theory of Distributive Fairness for Multiple Goods

The above formulation of the theory of equitable optimality relied on the availability of an objective, common measure of agent outcomes in the distribution problem. This reliance removed the need to consider subjective utilities. Yet, in some applications it is not possible to agree on a common, objective valuation of outcomes. For such applications, distributive fairness models may be formulated using subjective utilities.

The theory of equitable optimality requires another simplification – it considers only a single-good distribution problem. Of course, this difficulty could be partially removed by considering a bundle of goods that was evaluated as a unit; yet, in a multi-good distribution problem, one good may be traded off against another in an outcome.

In this section, we shall formulate a theory of distributive fairness for the multiple-good distribution problem, based on the work of [149]. The theory is in many respects similar to the theory of equitable optimality, which can be reformulated in the same setting using subjective utilities for multiple goods. Yet, equitable solutions (in the sense discussed above) are only a subset of the solutions obtained from the new theory of distributive fairness. For simplicity

of discussion, we shall refer to the new theory as *Generalized Distributional Fairness*.

As defined in section 2.3.2, let Y be the set of all possible outcomes (not only the efficient outcomes, but all) of the distribution problem. In the definition of the theory of equitable optimality, we assumed that there existed a common, objective, measurable criterion of evaluating the outcome of any agent. This criterion was not explicitly denoted, but rather it was assumed that outcome values y_i are in the units of that criterion. For a multi-good problem, $\mathbf{y}_i \in Y$ is a vector that describes how much of any good the agent i has received in outcome $\mathbf{y}$. Also, we are now going to consider the case where a common, objective criterion of outcome valuation does not exist. Rather, each agent values his outcome subjectively, using a unique utility function. This real-valued vector function will be denoted by $\mathbf{u}(\mathbf{y})$. We shall assume that agents aim to maximize their utility.

It has been noted previously that in order to introduce a concept of distributive fairness, there must be a way of comparing the outcomes of individual agents. This observation still holds (even more strongly) in the case of subjective utilities [6]. In order to introduce the theory of Generalized Distributional Fairness, it is necessary to assume that the subjective utilities can be compared. Yet, there can be many different kinds of comparison [95]. For our purposes, it is sufficient to assume that *the utilities of individual agents can be sorted in a non-decreasing order (from worst to best)*. Let us denote the vector of *ordered utilities of all agents* for outcome $\mathbf{y}$ by $u^*(\mathbf{y}) = [u_1^*(\mathbf{y}), u_2^*(\mathbf{y}), \ldots, u_n^*(\mathbf{y})]$ such that $u_1^*(\mathbf{y}) \leq u_2^*(\mathbf{y}) \leq \ldots \leq u_n^*(\mathbf{y})$. Thus, while u_i is the utility of agent i, u_i^* is the utility of the ith worst-off agent in the ODS.

Note here that ordered utility functions are not quite enough to use the theory of equitable optimality. The assumption that utilities are comparable is not equivalent to the assumption that the differences between utilities are comparable (the first assumes an ordinal relation on utilities, while the second assumes a cardinal relation). If, as required by the theory of equitable optimality, we are going to compare sums of utilities, it is necessary to use a stronger assumption of cardinal comparison. Then, it would be possible to apply the theory of equitable optimality to the problem of multiple goods with subjective utilities that has been described so far in this section. However, we shall present a different theory of fairness that is more general than the theory of equitable optimality. Then, we shall compare the new theory to the

[6] The famous Arrow's impossibility theorem is a consequence of the lack of ability of comparison of subjective utilities. The theorem shows that a fair solution to the problem of aggregating social opinions does not existunless the solution is imposed by a dictator. However, the theorem's assumptions do not apply if there is a way of comparing the agents' subjective valuations (utilities) of available alternatives. Under such conditions, it is possible to find a fair social opinion. A similar reasoning to Arrow's would apply to fair distribution problems if the subjective utilities of agents could not be compared.

theory of equitable optimality, also for the single-good problem with universal utilities that has been described so far.

The theory of Generalized Distributional Fairness, formulated in [95], but based on the work of Sen and Suppes [149], is defined as follows, using two concepts: the *Generalized Rawlsian Fairness relation of the kth degree*, and the *Generalized Conservative Fairness relation of the kth degree*.

For two possible outcomes $y^{'}$ and $y^{''}$, $y^{'}$ is considered as fairer than $y^{''}$ according to the Generalized Rawlsian Fairness relation of the kth degree, if and only if $u_i^*(y^{'}) \geq u_i^*(y^{''})$ for all i such that $1 \leq i \leq k$, and there exists at least one j such that $1 \leq j \leq k$ and $u_j^*(y^{'}) > u_j^*(y^{''})$. This condition is equivalent to Pareto-dominance of the sorted utility vector for the k worst utilities.

Similarly, for two possible outcomes $y^{'}$ and $y^{''}$, $y^{'}$ is considered as fairer than $y^{''}$ according to the Generalized Conservative Fairness relation of the kth degree, if and only if $u_i^*(y^{'}) \geq u_i^*(y^{''})$ for all i such that $n-k+1 \leq i \leq n$, and there exists at least one j such that $n-k+1 \leq j \leq n$ and $u_j^*(y^{'}) > u_j^*(y^{''})$. This condition is equivalent to strong Pareto-dominance of the sorted utility vector for the k best utilities among the utilities of n agents. The conservative preference relation can be thought of as favoring the fittest (and thus best-off) agents.

Let us define two sets of solutions in Y: the set R_k of the solutions that are optimal (maximal) with respect to the Generalized Rawlsian Fairness relation of the kth degree, and the set C_k of the solutions that are optimal (maximal) with respect to the Generalized Conservative Fairness relation of the kth degree. Notice that the sets R_n and C_n are equal (and equal to the set of all Pareto-optimal ordered solutions in Y). This set is also the set of solutions that are optimal with respect to the fairness relation proposed by Suppes [95], and will therefore be called the Suppes set, S. The solutions belonging to S are exactly the ordered Pareto-optimal solutions of the efficient optimization problem, as discussed in section 2.3.2.

Using these sets, we can now define a broad class of solutions that are optimal according to the theory of Generalized Distributional Fairness. These solutions are the sum of three sets: $S \cup GR \cup GC$, where $GR = R_1 \cup \ldots \cup R_{n-1}$ and $GC = C_1 \cup \ldots \cup C_{n-1}$.

If we assume cardinal comparison of agent utilities, then we can apply the theory of equitable optimality and obtain the set of its optimal solutions, denoted by O. We can further refine this notion by the following definition: let the set of O_k denote the equitable solutions of the problem $\max \{\theta_1(\mathbf{y}), \ldots, \theta_k(\mathbf{y})\}$, where $theta_k(\mathbf{y}) = \sum_{i=1}^{k} u_i^*(y)$. Then, $O = O_n$. We shall now show the relationship between the theory of equitable optimality and the theory of Generalized Distributional Fairness.

Theorem 2. *An outcome $\boldsymbol{y} \in Y$ that is equitably efficient according to the theory of equitable optimality is also optimal according to the theory of Generalized Distributional Fairness. In particular, $O_k \subset R_k$.*

Proof: If an outcome $\mathbf{y} \notin R_k$, then there exists an outcome $\mathbf{y}'$ that Pareto-dominates $\mathbf{y}$. This means that

$$u_1^*(\mathbf{y}') \geq u_1^*(\mathbf{y}), u_2^*(\mathbf{y}') \geq u_2^*(\mathbf{y}), \ldots, u_k^*(\mathbf{y}') \geq u_k^*(\mathbf{y}),$$

where at least one inequality also is strict. But then, by taking increasing sums of the inequalities we get:

$$\theta_1(\mathbf{y}') \geq \theta_1(\mathbf{y}), \theta_2(\mathbf{y}') \geq \theta_2(\mathbf{y}), \ldots, \theta_k(\mathbf{y}') \geq \theta_k(\mathbf{y}),$$

where also at least one inequality is strict. This means that $\mathbf{y} \notin O_k$, which implies that $O_k \subset R_k$. □

However, the reverse inclusion does not hold. To see why this is so, consider the example of a distribution problem for just two agents (assuming there is a single good, and the agents have similar utilities). This example was already discussed in section 2.3.2, and the solutions belonging to $R_2 = S$ (because $n = 2$) shown in Figure 2.6. Here, let us directly compare the dominating sets $S(y)$ for a solution y. The comparison is shown in Figure 2.9. The area in the middle of the figure with a finely dashed background belongs to the dominating set according to the theory of equitable optimality, but not according to the theory of Generalized Distributional Fairness. In particular, point y' dominates y according to the theory of equitable optimality, but not according to the theory of Generalized Distributional Fairness. If the set of all possible outcomes is given by: $Y = \{y, y', y''\}$, then $O_2 = \{y'\}$, but $R_2 = \{y, y', y''\}$. Thus, the example shows that the inclusion $O_k \subset R_k$ is strict.

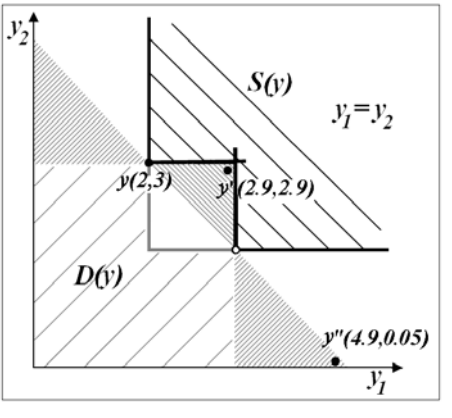


Fig. 2.9 Comparing equitable solutions with solutions of Generalized Distributional Fairness in 2D: $D(y)$ - set fairly dominated by y, $S(y)$ - set of outcomes fairly dominating y

Note that another way of showing that $O_n \subset R_n$ would be to observe that the set O_n is defined as the set of solutions optimal according to preference relations that conform to the three axioms of the theory of equitable optimality (as shown by theorem 1). However, R_n (the set of ordered Pareto-optimal solutions) is also the set of solutions optimal according to preference relations that conform to just two of these axioms (without the principle of transfers). Thus, solutions in O_n must also belong to R_n. The result shown above extends this observation to O_k and R_k, for $k < n$.

In this section, we have extended the theory of equitable optimality to a problem with multiple goods and subjective utilities by defining the set of $O_n = O$ of equitably optimal solutions for this problem. We have also introduced the Generalized Distributional Fairness (GDF) theory that extends the set of solutions considered fair. This extension is rather large, as the set of equitably optimal solutions is strictly included in just a part of the solutions of the theory of GDF. However, the extended theory has an important interpretation in the social sciences: it can be shown that the solutions of the theory of Generalized Distributional Fairness describe well the solutions chosen as fair by real human subjects in empirical experiments.

2.3.5 Game-Theoretic Notions of Fairness

Distribution problems in an ODS have a high resemblance to game-theoretic settings. The notion of an encounter itself is similar to a game. However, the conditions of an ODS are usually very complex, and many of the basic assumptions of game-theory are too limiting in practice. In fact, much empirical evidence today suggests that the most basic assumptions of game theory: that players are selfish; that players are strategic; and that an equilibrium can be reached, do not apply in practice to the behavior of human agents (see [25]). Still, concepts of game theory have frequently been applied in ODS, and have also lead to practical solutions. A well known example is the application of the Tit-for-Tat strategy for the iterated Prisoner's Dilemma in Bittorrent for the prevention of free-riding. The reasons for the usefulness of some game-theoretic concepts is that they have been developed for situations where centralized control is not available. In games, players make autonomous decisions, and this similarity to an ODS is sufficient for game-theoretic strategies to be applicable in practical ODS. It is also possible to consider game-theory as a pessimistic approximation of human behavior (since it assumes selfish and strategic behavior). In this section, we shall consider game-theoretic concepts that are useful in the theory of fairness.

2.3.5.1 Noncooperative Games

Non-cooperative games can be used to model a situation when selfish, strategic agents interact in encounters with no incentive to cooperate (there exists

no enforcement mechanism of fair behavior). An example of such a situation are peers who share files in a P2P overlay network. Every peer is interested to receive files, but is not interested to provide files to others. This situation is usually modeled using games which have two actions for each agent: $Cooperate$ and $Defect$. The structure of payoffs is usually such that the Nash equilibrium in a single game is the choice of the $Defect$ action by all players. The Nash equilibrium is clearly not the utilitarian optimal choice (a choice that would result in the highest sum of payoffs of all agents). For our example of P2P file sharing, the Nash equilibrium is the choice of never providing files by all peers in the system. As a result, no file would be available in the system for retrieval. The ratio of the highest sum of payoffs of all agents to the sum of payoffs in the worst Nash equilibrium is called the *price of anarchy*. In our example, the price of anarchy is infinite.

Note here that the concept of a Nash equilibrium in a noncooperative game should only be used if it can be proven that the game players are selfish. If they are not, the Nash equilibrium is not likely to be chosen. Empirical research from behavioral game theory supports the notion that Nash equilibria are rarely chosen by real players.

The most popular game used in noncooperative game theory is the Prisoner's Dilemma (PD) [7]. This game has in-built incentives for cooperation, although the Nash equilibrium of the game is still uncooperative. It can be said that research on the iterated PD and its famous conclusion about the evolution of cooperation has created a paradigm in the research on fairness. In this paradigm, fairness is identified with reciprocity, and the noncooperative game theory assumptions are considered true: agents are selfish, strategic, and uncontrolled. The PD allows us to measure cooperation using a simple utilitarian function (the sum of all agent's payoffs), as the sum of payoffs is highest for the cooperative solutions of the game. This method is also a utilitarian approach to the evaluation of trust management and reputation systems [118, 34, 178]. The payoff structure of the PD can be characterized as follows:

PayoffA,PayoffB	Cooperate	Defect
Cooperate	R,R	S,T
Defect	T,S	P,P

Fig. 2.10 Structure of payoff in the Prisoner's Dilemma

The payoffs of the game (R: Reward for mutual cooperation, S: Sucker's payoff, T: Temptation to defect, and P: Punishment for mutual defection) fulfil the following relations:

$$T > R > P > S,$$

and

$$T + S < 2R.$$

These relations guarantee that the Nash equilibrium is mutual defection, while the Pareto-optimal (also utilitarian optimal) solution is mutual cooperation. The values for the payoffs could vary, as long as the relations presented above are satisfied. By varying the payoffs, the PD could become more similar to a constant-sum game. The most frequently used payoff structure of the PD is as follows: $T = 5$, $R = 3$, $P = 1$ and $S = 0$.

There are many real-life examples of Prisoner's Dilemmas. One of the most well-known examples, described in Axelrod's book, is the trench warfare of World War I. Companies of soldiers located on two sides of the frontline would refrain from attacking each other, although a surprise attack could result in a significant gain. However, attacks were extremely dangerous for both sides; and the reward for mutual cooperation was high. Another frequently used example of the Prisoner's Dilemma is the arms race between countries in the Cold War.

A more general form of the Prisoner's Dilemma is the n-person PD, also called a *Social Dilemma* (SD) [82, 81, 74]. The SD differs from the PD in several respects. The negative effects of defection affect not one, but many agents involved in the encounter. Also, identifying defectors may be more difficult, because the perceived negative results of defection could be caused by any of the other agents.

Social dilemmas are further classified into resource dilemmas and public good dilemmas. Resource dilemmas describe a situation where a scarce resource can be harvested by many people. Every agent in a resource dilemma is tempted to take large amounts of the resource (defect), but if sufficiently many agents harvest too much, the resource is depleted for all other agents. Participants of the resource dilemma may keep their harvests as long as the resource is not depleted completely. In the case where the resource is depleted, all participants lose their harvests.

A public good dilemma models a situation where people contribute towards a public good (a good that benefits all agents without possibility of exclusion). The public good can only be established by a sufficient amount of contributions from agents, but once established, it is usually multiplied by a factor. Therefore, the possibility exists that some agents will not contribute to the public good, but will benefit from it (free-ride). On the other hand, if too many agents free-ride, then all agents will be deprived of the public good. Two stages can be distinguished in a public good dilemma: the first stage requires the provisioning of the public good. In this stage, it is necessary to motivate agents to contribute the required resources or bear the necessary costs. The second stage is the distribution or allocation of the public good. Depending on additional assumptions on the public good problem, the distribution stage may be ignored (if it is assumed that all agents have equal access to the public good once it is established). However, if it is possible to control access to the public good, it may be distributed among agents. The problem of its distribution is then a typical fairfair distribution problem, and the theory of equitable optimization may be applied.

2.3.5.2 Cooperative Games

The basic definition of cooperative game theory is that it is a game where players can enforce fair behavior. However, this definition can be relaxed. If the fair agreements made on the basis of cooperative game theory are violated, trust management can be applied to punish offending peers.

Cooperative game theory is concerned with the distribution of gains that a group of agents obtain from cooperation. The model assumes that the group of agents wishes to solve a common problem. An example could be the problem of waste disposal by cities: each city needs to solve a problem, therefore it is common to all. By cooperating, cities could dispose of their waste more efficiently. All agents that could potentially cooperate form a set A. The task of cooperative game theory is to determine what subset K of A would actually agree to cooperate in a given situation.

Cooperative game theory assumes the existence of a cost function, c (also called the "alternative costs"), that takes as an argument any subset B of the set A. The cost function returns the cost $c(B)$ for which the agents in B could solve the problem. B could contain only one agent. If B contains more than one agent, then the agents in B would still need to divide the cost among themselves, but how this is done is not specified by the cost function. Since agents are selfish, for every agent $a \in A$ the cost function determines an upper bound on the agent's share of costs in a cooperation: no agent would agree to contribute more than $c(a)$.

For every agent a, the cost function also determines the cost $c(A - a)$, which is the cost at which all other agents could solve the problem without the cooperation of a. This cost determines a lower bound on the share of costs of the agent a (the cost contribution of a). If the agent was willing to contribute less, it would be possible for all other agents to solve the problem without him at a lower cost. As a matter of fact, if $c(a) < c(A-a)$, it is clear that a would never be part of the cooperation with all other agents.

The same reasoning can be applied to any subset B of A. The reasoning of cooperative game theory leads to the concept of a set of cooperating agents that divide the costs among themselves in some way. The cost divisions of the cooperating agents must fulfil the property that for any subset of B (the cooperating agents), their cost contribution must be less than their alternative costs ($c(A) - c(A - B) < c(B)$). The set of this cost divisions is called the *core*. It is also possible to think of a set of cooperating agents that have a non-empty core. If the core is empty, then the considered set of agents will not cooperate in the game-theoretic model.

The core can still be a large (if the costs are continuous, even infinite) set of cost allocations among the cooperating agents. In order to suggest a single solution to the problem of cost allocation, cooperative game theory defines a new problem of distribution. Instead of looking at the costs, the theory now considers the gains (or savings) of each agent. For any cost allocation that

assigns the cost of c^a to agent a, the gains of a are $c^a - c(a)$. It is assured that this value is not negative if the cost allocation is part of the core.

The problem of finding a fair solution to the gains distribution problem in cooperative game theory is posed as the problem of fair distribution of savings for cost allocation inside the core. In other words, the core specifies constraints on the cost allocations, while the optimization criteria are the savings of cooperating agents. Cooperative game theory defines the concept of the *nucleolus*, which is a cost allocation in the core obtained by lexicographically maximizing the savings of all agents.

However, the reader will notice that it is possible to extend cooperative game theory using the theory of equitable optimality. The nucleolus need not be the only equitable solution of the problem of gains distribution in the core. It would also be possible to suggest methods that would search for other such solutions. The problem could be defined as a multicriteria optimization problem, and solved using various preferences of the decision makers.

It would also be possible to define the problem of searching for fair distributions inside the core as a problem of equitable distribution of costs, and not of gains. The formulation of the criteria of the problem could also be chosen by decision makers.

It follows that collaborative game theory and the theory of equitable optimality complement each other. The game-theoretic model allows us to take alternative costs that form a set of constraints on equitable solutions into account. On the other hand, equitable optimization methods can be used to search for various equitable solutions that satisfy these constraints.

2.3.5.3 Fair Negotiations

Fair negotiations can be used to solve distribution problems, or to arrive at fair agreements. The game-theoretic theory of fair negotiations assumes that all agents in an encounter have to agree on a particular solution to the distribution problem. If any agent disagrees, negotiations are broken and the encounter does not take place. It is assumed by game theory that in such a case, the amount of goods that could be distributed during the encounter is lost. Another assumption is that goods are divisible (if they are not, it is possible to divide the chances of obtaining the goods).

The game-theoretic negotiation model further assumes that the agents make their decisions based on expected utilities, and that these utilities will have a functional form that reflects the agent's attitude towards risk. The source of risk in the negotiations is the ability of any agent to "veto" the outcome, leading to the loss of the divided goods by all agents in the encounter. Classical negotiation theory predicts two possible fair negotiation solutions to the distribution problem. The first of them, referred to as the Kalai-Smorodinski solution, can be described as follows. Every agent should be indifferent to receiving his outcome in the *Kalai-Smorodinski solution* with

certainty, and to the participation in a lottery where each agent can receive the entire available amount of goods with a fixed probability.

The other fair solution of the distribution of negotiated goods is the *Nash bargaining solution*. This solution gives all agents outcomes with the property that no transfer between any two agents would be justified using the Nash criterion. Using the Nash criterion, a transfer from agent A to agent B is justified if the relative decrease of the utility of agent A is less than the relative increase of the utility of agent B. Notice that the Nash criterion avoids interpersonal comparison of utilities, as it only compares the relative changes of utilities. Also, notice that if agent A is more risk averse than agent B, then they will not receive equal outcomes in the Nash bargaining solution. The reason for this is that the utility function of agent A will increase at a slower rate with the increase of the outcome, than the utility function of agent B. Thus, the relative decrease of A's utility will be less than the relative increase of B's utility if their shares are initially equal.

Both the Kalai-Smorodinsky and the Nash bargaining solution are efficient solutions to the distribution problem, using the definition introduced in the previous section.

The classical, game-theoretic negotiation theory has serious drawbacks. The first drawback is a lack of consideration for context, for example with regard to the relative priorities of the negotiating parties. Imagine that negotiations concern the distribution of the residual capital of a firm, and that the negotiating agents are creditors of this firm. Clearly, the amount that the firm owed to each creditor should have an impact on the outcome. In classical negotiation theory, only the risk attitudes of negotiators will have an impact on the solution.

A more significant drawback is that empirical observations from experimental negotiations have shown that human agents seldom choose distribution solutions that are in agreement with game-theoretic negotiation theory. Rather, negotiators tend to select solutions that are influenced by some common norm of fairness. In other words, negotiators that agree on a norm of fairness apply this norm to find an acceptable distribution solution, rather than play a game of nerves as expected by game-theoretic negotiations.

2.3.6 Empirical Investigations of Fairness Preferences

This section provides a summary of the research on fairness preferences described in [95, 96]. The results of this research are especially relevant for the evaluation of the introduced fairness theories.

The presented results are based on an experiment that uses a problem of social choice in order to evaluate human preferences with respect to the rules of fair distribution. The problem of social choice can be defined as follows: given individual opinions from a group of agents, create a single social opinion

for the entire group. A good example of the problem of social choice is the problem of determining the results of an election from the individual votes. In the experiment used to test fairness preferences, participants were asked to determine the verdict of a jury given the opinions of three jurors. The three jurors ranked three alternatives in a complete, weak order. Out of the three opinions, experiment participants had to create a jury verdict.

Such a problem of social choice can be reduced to an efficient multicriteria optimization problem by considering the distances of the individual agents' opinions to the social opinion. In the case of the three jurors, it is possible to compute a distance of every juror's opinion to the jury verdict. A distance measure that can be used for this reduction is the Kemeny-Snell measure (for details, see [77]). Naturally, it makes sense to require that the jury verdict should be fair in the sense that the distances of the individual jurors' opinions to the verdict should be fairly distributed. Therefore, a social choice experiment can be used to test the preferences of human subjects with respect to fair distribution. Additionally, the jury problem provides an attractive and simple experimental setting.

Each experiment participant has been asked to provide a jury verdict for 8 sets of jury opinions (jury profiles). A sequence of jury verdicts created by a single participant will be referred to as an *empirical strategy* of a participant. There were 140 participants, and therefore 140 empirical strategies. The question considered in the analysis of the experiment results has been: *which of the theories of fair distribution best explains the empirical choices of experiment participants?* A theory of fair distribution should create for each jury profile a set of solutions that correspond to possible jury verdicts. The theory explains an empirical strategy if each of the empirical choices of a participant made for a jury profile is in the set of possible jury verdicts predicted by that theory for this jury profile.

The empirical strategies of participants turned out to be widely distributed. Over 62% of the empirical strategies were unique. The analysis of the experiment results considered several possible methods of fair social choice and of fair distribution. For example, the jury's opinions were created using the majority method, the ranking method, and Borda's elimination method. However, each of the single, classical methods had a very limited capacity for explaining empirical choices. The best of the classical theories (the ranking rule) explained less than 11.5% of the empirical strategies. All the classical theories considered together explained less than 52% of the empirical strategies (even under the condition that a different theory could be applied to each of the 8 jury profiles).

In contrast, the theory of Generalized Distributional Fairness introduced in section 2.3.4 explained over 80% of the empirical strategies. This result demonstrates that the theory of Generalized Distributional Fairness defines a very broad class of fair solutions that corresponds well to the fairness preferences of human subjects. Notice that the theory of equitable optimization defines a smaller subset of this set of fair solutions. Yet, both theories are

posed as multicriteria problems, and multicriteria optimization methods can be used to select solutions that best match the preferences of a decision maker. Using one of the two theories, a decision maker will be searching in a set that most likely includes his fairness preferences. Equitable solutions from the theory of equitable optimality have the additional property of being more easily computable than solutions of the Generalized Distributional Fairness theory.

2.3.7 Cooperation and Reciprocity

Following the work of Axelrod [7], a large body of research has considered the emergence of cooperation in the iterated Prisoner's Dilemma. The introduction of reputation has been demonstrated as helpful to the emergence of cooperation.

The problem of cooperation is often similar to the Social Dilemma problem introduced in section 2.3.5.1. In most research on this problem, it has been assumed that participating agents are selfish (choose options with the most beneficial outcome to themselves, without considering outcomes of others). However, research in psychology has revealed that in group situations, the decisions of individuals are influenced by motives such as group performance, sense of responsibility for others, or social concern. For a good overview of research results on this subject, the reader is referred to the work of Kazemi [74].

Research in psychology [74] has also revealed that in fair distribution problems in group settings (fair allocation of public goods), agents revealed preferences for equitable treatment that took into consideration inputs of agents, rather than a simple equal division of the public good. Furthermore, when the group needed to reach a goal of economic productivity, agents preferred an equitable distribution of the public good that rewarded agents who had contributed more to the establishment of the public good. On the other hand, if the group goal was social harmony, then agents preferred a distribution according to equal final outcomes (that took into account a difference between the contribution and the value of the received public good. This approach is similar to the use of a utility function that is equal for all agents).

2.3.8 Economic Notions of Fairness

The theory of distributive fairness described in this chapter does not assume that an agents' utilities can be explicitly expressed. Economic theories of fair allocations, on the other hand, usually rely on the reverse assumption, that an agents' utilities can be expressed as computable functions or relations. Such theories are built on the basic concept of no-envy that relies on the ability of each agent to compare a received outcome with any other outcome received by another agent. This comparison is not based on objective outcome criteria

(such as amounts of goods, money or services), as in the theory of distributive fairness. Rather, they are based on subjective preferences of agents.

Economic theories of fairness [161] combine the notions of Pareto-optimality and no-envy. A solution is deemed fair under economic theory of fair allocation, if it is Pareto-optimal and it is a no-envy solution. Such solutions exist, and a good example which is a special type of a Pareto-optimal, no-envy solution, is the *equal-division Walrasian allocation*. The Walrasian allocation can be defined as the allocation that maximizes each agent's utility for any good, subject to a total budget constraint that is the same for all agents. The budget constraint is the total cost of a bundle of goods that is an equal division of every good between all the agents. The total cost is calculated for a set of common (usually normalized) prices among all agents.

Equal-division Walrasian allocations are not the only Pareto-optimal and no-envy allocations, but they can be thought of as the result of a market process. In a perfect, stable market, every agent can maximize his subjective utility by purchasing the goods he wants, using the budget obtained from an equal-division initial allocation of goods (that he can sell on the market). The equal division Walrasian solution also has interesting theoretical properties: in the theory of mechanism design, this solution has a minimal dimension of message spaces required for its realization, and therefore has the highest informational efficiency. Moreover, it can be shown to be the only such mechanism [24]. Another important feature is that this mechanism only depends on local information about agent preferences. Other mechanisms for determining no-envy solutions cannot operate based only on local information.

Economic theories of fairness distinguish between the fairness of initial allocations, the fairness of trades (or transitions between allocations), and the fairness of the final allocation (in a stable market solution). Equal division is the usual choice of a fair initial allocation. The criteria of fair trades are usually also based on the principle of no-envy. The final allocation ought to be a Pareto-optimal, no-envy allocation. However, these considerations lead to certain paradoxes [161]. For example, when starting from an initial Pareto-optimal, no-envy allocation, and applying no-envy trades, one can obtain an allocation that is not envy-free. Also, there may be a non Pareto-improving trade that leads to a particular Pareto-optimal, no-envy solution, starting from a certain initial allocation. The only fairness principles that are consistent with respect to transitions are the Walrasian principles. Starting from an equal-division Walrasian allocation, Walrasian trades always lead to another equal-division Walrasian allocation.

To conclude the discussion of economic notions of fairness, recall that these notions always assume the existence and computability of subjective agent preferences. Furthermore, the most appealing solution that satisfies economic notions of fairness (i.e., is Pareto-optimal and envy-free) is the equal-division Walrasian solution that assumes the existence of a perfect market.

Finally, let us give an example that demonstrates a basic problem of the economic notions of fairness. Consider a set of agents that are animals in a zoo.

These animals have utilities that can be expressed by the amount of calories that they consume, but they can only consume certain types of food. Thus, their utilities are subjective. The problem of fair distribution of food among these animals can be solved by applying economic notions of fairness. Yet, consider an animal that can only eat one type of food that is also eaten by certain other animals – for example, eggs. A no-envy, Pareto-optimal solution for the zoo can give the egg-eater an insufficient amount of eggs for him to survive. However, it is both Pareto-optimal and envy-free, as giving the egg-eater more eggs would require taking them away from some other animals that would also need them, and the egg-eater already has more eggs allocated than any other animal. The weakness of the concept of no-envy is demonstrated in this example: no-envy does not take into account the actual level of utility achieved by the agents that are worst off. This level of subjective utility can actually be much lower than that of the other agents (in our example, below starvation level). The theory of equitable optimality starts with the consideration of the improvement of the utility of the worst-off agent.

A connection of the economic notions of fairness and the theory of equity can be made by requiring that the two theories are consistent if the subjective utilities of agents are the same. In the case of equal utilities of all agents, Pareto-optimal, no-envy solutions are reduced to equal distributions. A generalization of the no-envy principle can lead, for agents with equal preferences, to other equitable solutions.

2.3.9 Computational Approaches to Distributive Fairness Problems

There can be many approaches for obtaining computational solutions of distributive fairness problems. All of them would somehow solve the efficient optimization problem. If by distributive fairness we will understand equity, then a computational approach needs to solve the equitable optimization problem (1). Recall that this was a multicriteria optimization problem obtained from the original problem of efficient optimization by taking the ordered sums of the outcomes, starting from the worst off, then the sum of the worst and second-worst, and so on.

An intuitive approach for a fair solution to this problem would be lexicographic optimization. In other words, consider the first criterion that is worst-off and optimize it; then take the sum of the worst and second-worst, and so on until we consider the sum of all criteria. This approach gives, however, only one of the many possible Pareto-optimal solutions of the equitable optimization problem. In some sense, it is an extreme solution – it puts the most emphasis on equality. If there are no constraints, the lexicographic optimization will always find a perfectly equal distribution.

While the lexicographic optimization is only a specific solution concept, the entire multi-criteria problem (1) may serve as a source of various equitable solutions [127]. Moreover, it may be modeled using linear auxiliary

constraints. Thus, if the original efficient optimization problem was linear, the equitable optimization problem is also in the computational class of linear optimization problems [126].

Let us notice that for any given vector $\mathbf{y}$, the quantity $\theta_k(\mathbf{y})$ is defined by the following LP problem:

$$\begin{aligned} \theta_k(\mathbf{y}) \;&=\; \min \sum_{i=1}^{n} y_i z_i \\ &\text{s.t.} \sum_{i=1}^{n} z_i = k,\; 0 \le z_i \le 1 \quad \text{for all } i. \end{aligned} \tag{2.6}$$

The above problem is an LP for a given outcome vector $\mathbf{y}$ while it becomes nonlinear for a variable $\mathbf{y}$. This difficulty can be overcome by taking advantage of the LP dual to (2.6):

$$\begin{aligned} \theta_k(\mathbf{y}) \;&=\; \max\; kt - \sum_{i=1}^{n} d_i \\ &\text{s.t. } t - y_i \le d_i,\; d_i \ge 0 \quad \text{for all } i, \end{aligned} \tag{2.7}$$

where t is an unrestricted variable while nonnegative variables d_i represent downside deviations of outcome values y_i from the value of t [126].

Formula (2.7) allows us to formulate problem (1) as follows:

$$\max\,(\theta_1, \theta_2, \ldots, \theta_n) \qquad \text{subject to} \qquad y \in Q, \tag{2.8}$$

$$\theta_k = kt_k - \sum_{i=1}^{n} d_{ik} \quad \text{for all } k, \tag{2.9}$$

$$t_k - d_{ik} \le y_i, \quad d_{ik} \ge 0 \quad \text{for all } i, k\,. \tag{2.10}$$

Note that problem (2.8) only adds linear constraints to the original attainable set Q. Hence, if the basic optimization problem with the set Q was in LP, the resulting formulation (2.8) remains in the same computational class.

2.3.9.1 Solving Multi-criteria Distributive Fairness Problems

A computational approach to distributional fairness needs to give a solution to the equitable distribution problem. Since this problem has been formulated as a multi-criteria optimization problem, a solution can be found by applying one of the many known multi-criteria optimization methods. In this section we give a brief overview of a few such methods. The subject of this book is not multi-criteria optimization: the interested reader should refer to [175].

Before introducing some methods of solving multi-criteria problems, it is necessary to explain that solving such a problem can mean two things. Firstly, it can mean finding one of the Pareto-optimal solutions. Secondly, it can mean finding the entire set of all Pareto-optimal solutions. The second

task is often very difficult. On the other hand, the first task may be more simple, but the question remains as to which solution is the right one. Multi-criteria analysis and decision making solve this problem by assuming that the decision maker is able to specify his preferences, sometimes in an interactive procedure. Preferences are expressed in a way that allows them to be used by the chosen multi-criteria optimization method. These preferences allow us to find a unique Pareto-optimal solution.

There can be many ways of expressing preferences. One of the simplest of these is using weights. For the ordinary weighted sum, it can be shown that no matter how the preferences (weights) are specified, optimizing the weighted sum may not discover all available Pareto-optimal solutions (for non-convex sets of all Pareto solutions). On the other hand, another method is available that seems uniquely suited to the theory of equitable optimality. This is the Ordered Weighted Average (OWA) [185].

The OWA method can be introduced by considering the ordering of outcomes from worst to best, as in the theory of equitable optimality. Let these ordered outcomes be denoted as $y_{\pi i}$, where π_i is the index of the ith worst outcome. Then, the OWA method can be defined as:

$$\max \left(\sum_{i=1}^{n} w_i y_{\pi_i} \right).$$

If the weights w_i are strictly decreasing ($w_1 > w_2 > \ldots > w_m$), each optimal solution corresponding to the OWA maximization is a Pareto-optimal solution of (2.8). Hence, each optimal solution of the OWA maximization with strictly decreasing weights is an equitable solution of the efficient optimization problem.

The reason for this fact is that any equitable transfer results in a larger value of the OWA aggregation with strictly decreasing weights. On the other hand, an equitable transfer within a class of equal weights w_i does not change the value of the corresponding OWA aggregation. This implies that the equity of OWA solutions can still sometimes be improved. Still, for LP efficient optimization problems, every equitable solution can be found as an OWA optimal solution with appropriate strictly monotonic weights [83]. Several decreasing sequences of weights allow us to find various equitable solutions, including solutions resulting from classical fairness models.

Still, weights are not an easy way to express decision maker preferences or to manipulate the selection of equitable solutions. Better controllability and the complete parameterization of nondominated solutions even for non-convex, discrete problems can be achieved with the direct use of the reference point method. The reference point method (RPM) is an interactive technique where the decision maker specifies preferences in terms of aspiration levels (reference point), i.e. by introducing desired (acceptable) levels for several criteria.

The reference point method was introduced by Wierzbicki [174] and later extended leading to efficient implementations of the so-called aspiration/

reservation based decision support (ARBDS) approach with many successful applications [175]. The ARBDS approach is an interactive technique allowing the DM to specify requirements in terms of aspiration and reservation levels, i.e., by introducing acceptable and required values for several criteria. Depending on the specified aspiration and reservation levels, a special scalarizing achievement function is built which may be directly interpreted as expressing the outcomes to be maximized. Maximization of the scalarizing achievement function generates an efficient solution to the multi-criteria problem. The solution is accepted by the DM or some modifications of the aspiration and reservation levels are introduced to continue the search for a better solution. The ARBDS approach provides a complete parameterization of the efficient set of the multi-criteria optimization problem. Hence, when applying ARBDS methodology to the ordered cumulated criteria in (2.8), one may generate all (fairly) equitable solutions of the original efficient optimization problem.

While building the scalarizing achievement function the following properties of the preference model are assumed. First of all, for any individual criterion η_k (in our case, the cumulative sum of outcomes) more is preferred to less (maximization). To meet this requirement the function must be strictly increasing with respect to each outcome. Secondly, a solution with all individual outcomes η_k satisfying the corresponding reservation levels is preferred to any solution with at least one individual outcome worse (smaller) than its reservation level. Next, provided that all reservation levels are satisfied, a solution with all individual outcomes η_k equal to the corresponding aspiration levels is preferred to any solution with at least one individual outcome worse (smaller) than its aspiration level. This means that the scalarizing achievement function maximization must ensure reservation levels are reached prior to further improvement of criteria. In other words, the reservation levels represent some soft lower bounds on the maximized criteria. When all these lower bounds are satisfied, the optimization process then attempts to reach the aspiration levels.

The generic scalarizing achievement function takes the following form [174]:

$$\sigma(\eta) = \min_{k \in K} \{\sigma_k(\eta_k)\} + \varepsilon \sum_{k \in K} \sigma_k(\eta_k), \tag{2.11}$$

where η_k are the scalarized criteria, ε is an arbitrary small positive number and σ_k, for $k \in K$, are the partial achievement functions measuring actual achievement of the individual outcome η_k with respect to the corresponding aspiration and reservation levels (η_k^a and η_k^r, respectively). Thus the scalarizing achievement function is essentially defined by the worst partial (individual) achievement but additionally regularized with the sum of all partial achievements. The regularization term is introduced only to guarantee the solution efficiency in the case where the maximization of the main term (the worst partial achievement) results in a non-unique optimal solution.

The partial achievement function σ_k can be interpreted as a measure of the DM's satisfaction with the current value (outcome) of the k-th criterion. It is a strictly increasing function of outcome η_k with value $\sigma_k = 1$ if $\eta_k = \eta_k^a$, and $\sigma_k = 0$ for $\eta_k = \eta_k^r$. Thus the partial achievement functions map the outcomes values onto a normalized scale of DM's satisfaction. Various functions can be built meeting these requirements [175]. We use the piecewise linear partial achievement function introduced in [120]. It is given by

$$\sigma_k(\eta_k) = \begin{cases} \gamma(\eta_k - \eta_k^r)/(\eta_k^a - \eta_k^r), & \text{for } \eta_k \leq \eta_k^r \\ (\eta_k - \eta_k^r)/(\eta_k^a - \eta_k^r), & \text{for } \eta_k^r < \eta_k < \eta_k^a \\ \beta(\eta_k - \eta_k^a)/(\eta_k^a - \eta_k^r) + 1, & \text{for } \eta_k \geq \eta_k^a, \end{cases} \tag{2.12}$$

where β and γ are arbitrarily defined parameters satisfying $0 < \beta < 1 < \gamma$. This partial achievement function is strictly increasing and concave which guarantees its LP computability with respect to outcomes η_k.

The aspiration and reservation values for the criteria can be selected interactively by the decision maker. As their selection is made separately for each criterion, it is easier to express decision maker preferences than by using weights (where the criteria needed to be compared against each other). For the equitable optimization problem, the criteria are cumulative ordered sums. For this reason, it is possible to select reservation and aspiration values using simple increasing sequences, which allows for a simple parametrization at the expense of the full power of the reference point method.

2.3.9.2 Solving Distributive Fairness Problems with Entitlements

In section 2.3.3 the distributive fairness problem was extended with the addition of entitlements that were modeled using positive integer weights v_i (the normalized weights have been denoted by $\overline{v}_i$). The problem with entitlements can be reduced to distributive fairness problems without entitlements by "cloning" agents so that each agent can have a weight of one. The outcomes for the problem with entitlements can be obtained by adding the outcomes of the "cloned" agents. However, as mentioned in section 2.3.3, this reduction – while theoretically valid – is inefficient, resulting in problems of dramatically increased size. It is therefore desirable to find computational methods that are able to find equitable solutions to the distributive fairness problems with entitlements directly, without having to reduce it to an unweighted problem.

The OWA method introduced in the previous section can be extended to achieve this purpose. The so-called Weighted OWA (WOWA) method [164] can be formulated as follows:

$$\max \left(\sum_{i=1}^{n} \omega_i y_{\pi_i} \right),$$

where

$$\omega_i = \omega^* \left(\sum_{k=1}^{i} \overline{v}_{\pi_k} \right) - \omega^* \left(\sum_{k=1}^{i-1} \overline{v}_{\pi_k} \right),$$

and ω^* is a piecewise linear function that interpolates points $(\frac{i}{n}, \sum_{k=1}^{i} w_k)$ together with $(0, 0)$.

Thus, the Weighted OWA formulation requires the same weights as the OWA method (w_i), but will also use the entitlement weights of agents, v_i. The WOWA method follows the rule that entitlements weights define a repetition measure within the distribution of outcome values while the OWA weights w_i are applied to averages within specific quantiles of size $1/n$ for this distribution.

The WOWA aggregation function can also be formulated similarly to the OWA function, with the OWA weights w_i applied to conditional means calculated according to the importance weights v_i instead of the original outcomes. For more details, see [122].

Notice that the WOWA method searches for equitable solutions for distributive fairness problems with entitlements by focusing on the quantiles of outcome distributions. The same approach can be applied in distributive fairness problems with a large number of participating agents. Using quantiles reduces the size of the problem dramatically, while still allowing us to increase the equity of resulting solutions.

2.4 Procedural Fairness

The previous section covered the theory of distributive fairness that can be applied whenever it is possible to precisely define a fair distribution problem and to find a solution that is accepted by the participants (or proposed and enforced by an authority). However, there remains the possibility that the distribution problem may be very complex or hard to define (for example, in the case of real estate, the legal rights to distributed goods may be complex or unclear), and the possibility that participants may not agree on a solution (such a possibility is already foreseen by game theory, as described in the previous section).

Procedural fairness is concerned with the fairness of decision procedures leading to outcomes even in the cases that have been described above. The entire legal system may be viewed as a system of procedural fairness operating in human societies. Systems of fair procedures are applied in dispute resolutions, or in cases when participants cannot agree on a solution of a distribution problem. Often, fair procedures require the introduction of *trusted third parties* and a distribution of control among participants. For this reason, procedural fairness is directly related to trust management.

Games (computer games or real games) are an interesting example of a situation where it is not possible to define a distributive fairness problem, yet if the goal of procedural fairness is realized, a solution exists that will be considered fair. This solution can be any game outcome if the established fair procedures (game rules) are not violated. Consider also that this example can be taken much further. It is possible to view any security policy as a set of rules that must be followed by all agents in the system. Therefore, problems of assuring security can be solved by procedural fairness systems if these systems can guarantee that the rules are obeyed or that misbehavior is quickly detected and can be punished.

Procedural fairness systems also find applications in cases where a distributive fairness problem can be solved, but the resulting distribution cannot be enforced, because the agents involved are autonomous and there is no trusted central control (as in an ODS). Consequently, the goal of procedural fairness is to design a system of procedures that leads to the enforcement of a fair or equitable solution.

According to Thibaut and Walker [162], procedural fairness can be accomplished using two types of control: process control and decision control. Process control gives agents control over the decision making process: agents can present evidence or arguments in favor of their preferred outcomes. Decision control gives agents direct control over outcomes. People generally prefer decision control over process control [162], but decision control is not feasible in all applications. Therefore, process control should be designed in such a way that it allows us to reach fair outcomes. Recent research in psychology [74] has found that procedural fairness is relational in character, and that people care about the results of fair procedures because these results reveal their own social position. Because of this, issues of neutrality, trustworthiness, and status recognition are central to understanding procedural fairness.

2.4.1 Fair Division Procedures

The problem of fair distribution can also be approached from the point of view of fair procedures. The goal of such procedures is finding a fair solution that would satisfy all participants in a way that does not require a central controller that would determine and impose a solution on all others. Fair division procedures attempt to satisfy fairness criteria that have been discussed in this chapter:

- proportional division (each agent receives not less than a share that is proportional to the inverse of the number of agents),
- envy-free division,
- efficient (Pareto-optimal) division,
- equal valuation division (all agents receive such shares that their subjective valuations of received shares are equal.

Many diverse procedures have been proposed that satisfy various fairness criteria. An additional criterion for such procedures is being strategy-proof. This criterion means that a rational who that is performing the division at any step of the procedure should not be able to influence the outcome in a way that is beneficial for himself. Such strategies usually require that the agent should have some knowledge of other agents' preferences and be able to falsely represent his own preferences. However, without assuming that such knowledge is available, the known fair division procedures usually fail to satisfy the criterion of efficient division.

The main drawback of known fair division procedures for more than two agents is that they usually fail to satisfy the criterion of Pareto-optimality. A majority of the procedures satisfies proportional division or envy-free division criteria. A good overview of fair division procedures is given in [17]. One of the best-known procedures described in the cited this book is the *adjusted winner procedure*. The procedure is a variant of the *Knaster procedure of sealed bids*. In these procedures, agents bid for the goods, and agents with the highest bids win. After the auction, the outcomes of agents are equalized. In the Knaster procedure, this is achieved by monetary payments. In the adjusted winner procedure, the goods are divided (which is only possible for divisible goods). For two agents, the adjusted winner procedure results in the Kalai-Smorodinski solution discussed in section 2.3.5.3. The Knaster procedure results in a proportional division but is not strategy-proof. Pareto-optimality is achieved by the *Gap procedure* [18]. This procedure also assumes payments for goods, and maximizes the sum of received bids. Then, the prices are reduced proportionally to the gap between the highest and next-highest bids. The results of this procedure are the prices of goods, and the division is done according to these prices. The Gap procedure also satisfies the criterion of proportional division. The allocations of the Gap procedure are not envy-free.

Procedures that would be able to find equitably optimal solutions in the sense discussed in this chapter are not known. Moreover, many of the procedures discussed here assume some measure of trusted central control – for example, in a procedure that assumes the payment for goods and the redistribution of surplus, the payments must be delivered to a trusted agent. Therefore, many of the proposed procedures are not well suited to a completely distributed implementation. In section 4.3, we shall show how the use of a Trust Management system can improve distributional fairness (in the sense of the theory of equitable optimality) in a completely distributed environment and in the presence of adversaries. However, the proposed solution does not provide any guarantees that the outcomes of agents will be equitably optimal.

Chapter 3
Trust Management

The user is sovereign.
Andrzej Wierzbicki

Trust, but verify.
Ronald Reagan

Trust Management (TM) is an area of information technology that aims to improve the operation of open, distributed systems by predicting or influencing the behavior of their users. When applied to human users, Trust Management methods attempt to leverage the human capacity for trust or distrust. Alternatively, TM systems can also be applied to control the behavior of non-human agents. In such a case, TM algorithms can support the fully automated control of agent operations, and have an impact on emergent properties of the ODS.

In this chapter, we will first consider the possible applications of TM systems such as Internet auctions and e-commerce systems, grid systems and other systems that can use the Web services architecture, and Peer-to-Peer systems. Next, we shall introduce a general model of TM systems that aims to become a unifying approach for the subsequent discussion of various TM algorithms. We shall devote some attention to the criteria of evaluation of TM systems: their correctness, effectiveness, and computational complexity. An important aspect of evaluation of TM systems is an adversary model that defines the possible behavior of agents in an ODS that aim to subvert or bypass the TM system. Adversary models will be used in the discussion of TM algorithms, but also in the next chapter that concerns Fairness Management.

This chapter shall then describe the various methods of expressing human trust in TM systems, in other words, ways of representing computational trust. The question here is not just of a choice of representation, like the choice of a continuous or discrete scale and of the ranges of this scale. A computational trust representation is tightly related with processing methods that are later the building blocks of more complex TM algorithms. These processing methods can be defined as operators on computational trust. In

A. Wierzbicki: Trust & Fairness in Open, Distributed Sys., SCI 298, pp. 71–143.
springerlink.com

section 3.5, we shall propose two new computational trust representations for two different applications: Internet auctions and recommender systems. The new computational trust representation for Internet auction is based on a detailed empirical analysis of Internet auction traces that allows us to determine how users represent and evaluate information relevant to trust management.

In section 3.6, we discuss various algorithms of trust management, such as algorithms for determining initial trust when the Trust Management system has no information about an agent, ways of gathering proofs, and computational trust propagation algorithms. Propagation algorithms fulfil a very important task in a TM system: based on available information, they can produce a new trust recommendation for a previously unknown agent. Such algorithms can also rank agents on the basis of trust or extend trust networks as widely as possible by recommending new trust relations. In this chapter, we present CloseLook, a highly efficient algorithm that can significantly reduce the computational and communication complexity of computational trust propagation [181].

The chapter concludes by a discussion of algorithms for calculating reputation in Internet auctions and P2P networks.

3.1 Applications of Trust Management

3.1.1 Internet Auctions and E-commerce Systems

The Internet economy is doing very well. According to Forrester Research and eMarketer, the e-commerce retail market is steadily growing with annual gains reaching 11% in 2009 (despite the global crisis) and 13% in 2008. Growth is broad-based and distributes almost equally among all categories of retail, travel and entertainment. Projection for the future is optimistic: annual gains will continue to grow at a double-digit level, reaching retail revenues of $156 billion in 2009 in the United States alone. Online auctions are among the most popular and important e-commerce services. It is estimated that over 15% of all e-commerce sales can be attributed to online auctions. EBay, the global leader in online auctions, has over 85 million active users (and hundreds of millions of registered users). Annual transactions on eBay surpass %50 billion. This immense marketplace for customer-to-customer e-commerce provides means for anonymous and geographically dispersed users to seal retail transactions. But an important question arises: how does one estimate the reputation of an anonymous business partner and how does one develop trust when there is hardly any history of business contacts between any two partners?

A reliable reputation system is crucial for enabling a fair and credible environment for e-commerce activities. The quality of the reputation system directly affects the credibility of an online auction service and impacts the amount of fraud present on the online auction market. Unfortunately, fraud

is still the main factor hindering further development of online auctions. According to the National Fraud Information Center in 2005, online auctions accounted for 42%[1] of all registered complaints with an average loss of $1,155. The number of complaints has grown quickly (12,315 complaints in 2005 compared to 10,794 in 2004) as well as total losses ($13,863,003 in 2005 compared to $5,787,170 reported lost in 2004). Online auction fraud definitely outranks other popular types of scams (in 2007, online auction fraud ranked third in NFIC rankings. Other e-commerce retail fraud was second.). As a result, the reputation system used by an online auction site must be robust enough to safeguard the online community of auction participants against fraud.

Devising a robust and fraud-free reputation system for auctions is difficult for various reasons. Most importantly, the reputation system must take into consideration high asymmetry between buyers and sellers in online auctions. These two classes of auction participants are exposed to different types of risk. Sellers are almost never threatened financially, because they can postpone the shipment of the merchandize until the payment is delivered. Therefore, sellers are generally not concerned with the reputation of their business partners. On the other hand, buyers decide upon participation in an auction solely based on the reputation of a seller. Furthermore, after delivering the payment, buyers are still in danger of receiving no merchandize, or of receiving merchandize of a lower quality and one inconsistent with the initial offer. From a buyer's point of view, a credible estimation of a seller's reputation is indispensable for secure and successful trade.

3.1.2 Web Services, Virtual Organizations and Grid Systems

Grid systems often use Web Services to implement frameworks of grid services. The Web Services architecture includes a special Trust Management method that relies entirely on recommendations in order to establish trust, and does not calculate reputation. The method also relies on trusted third parties: authorities that can issue recommendations to agents who wish to use a Web (or grid) service.

In the Web Services TM approach, an agent obtains certain recommendations (called security tokens) after authentication from an authority. When the agent wishes to invoke a Web Service, it will present his recommendations. The process of obtaining and presenting recommendations is standardized by the WS-Trust standard.

On the other hand, an agent that provides a service (for example, a member of the grid) can express his preferences concerning recommendations. Various recommendations can be required in order to have access to various services. The requirements of a providing agent can be specified using rules

[1] This number is grossly underestimated due to eBay's reluctance to cooperate with NFIC. NFIC estimates that the real number is closer to 70%.

described in the WS-Policy standard. When a requesting agent presents recommendations, the providing agent checks whether his policy is satisfied, and based on this check makes the decision of whether to provide the requested service.

The TM method used in Web Services and grids can be exploited to support workflows, yet this requires better methods for trust negotiation and retrieval of trust recommendations [14]. TM methods can also be used for access control in Grid systems [29].

3.1.3 Peer-to-Peer and Ad-Hoc Networks

In Peer-to-Peer systems and ad-hoc networks, the most common method of trust management is reputation. These systems are examples of the most decentralized ODS. Therefore, applied TM algorithms must be distributed.

Every designer of a P2P reputation system has to cope with the following challenges:

- How to establish trust between different peers without trusted third parties or authorities?
- How to gather and evaluate proofs?
- How to deal with adversaries in open P2P systems?

Section 3.7.3 gives an overview of P2P reputation systems.

3.2 Universal Trust Management

3.2.1 Trust Management as a Service

In the future, trust management may become yet another, standard service of information security, such as authentication, authorization, privacy or integrity. For this to happen, it is necessary to define standard primitives of trust management, and agree about what is in common among the many different applications, frameworks, architectures and languages of trust management studied to date. Currently, although there are many practical trust management systems, they are applied in widely different domains: some of them have centralized components, others are fully distributed; some use reputation, while others rely on recommendations for transferring or delegating trust. This makes it difficult to propose a unified approach to managing trust. On the other hand, generic architectures such as WebServices use trust management already.

In this section, a blueprint for a library of universal Trust Management services is presented and discussed. *A challenge in the design of the library is to make it sufficiently general, yet not too abstract.*

When Trust Management (TM) is treated as a service, its interfaces must be sufficiently general to support a variety of applications. In this section (and

in Figure 3.1), we present a generic scenario of an application that uses TM services. We introduce this scenario on two example applications: an Internet auction system and a Web service application. In the first case, the "agent" depicted in Figure 3.1 is the auction system itself. In the second example, the agent is an entity that invokes the Web service.

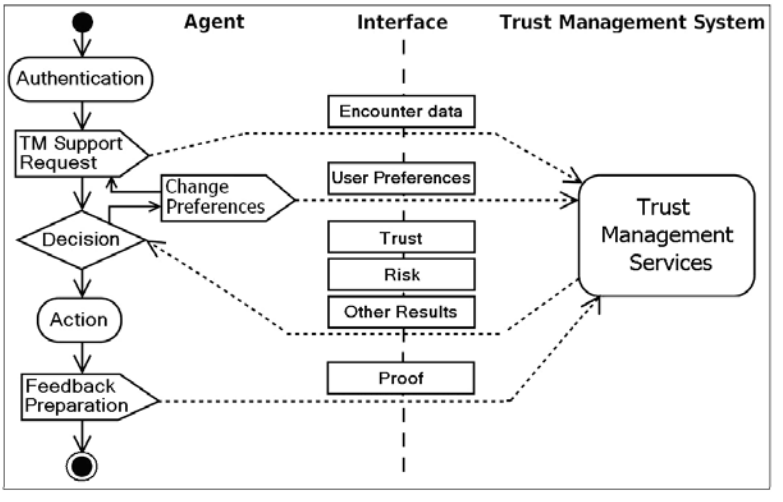


Fig. 3.1 Universal TM Services and Interfaces

The scenario begins with an authentications phase. Nothing is assumed about the authentication mechanism, although its quality will have an impact on the effectiveness of various TM algorithms and protocols. In the case of the Internet auction system, authentication may be based on a simple pseudonymous login and password. In the case of the Web service, there may be certificate-based authentication.

3.2.2 Universal Encounter Description

An agent that requests the help of TM service passes an the description of the future interaction, called the Encounter. An Encounter (see section 2.1) models all possible interactions between agents or applications that use TM services.

Relevant information about encounters should include agent identities, context, the actions available to agents and their outcomes. In section 2.1, context was defined as metainformation about the encounter. Context can be modeled as an arbitrary set of attributes that describe the encounter.

In the case of an Internet auction, encounter data includes the identity of a user (buyer or seller), and possible actions: sending of purchased goods,

or not sending them; paying or not paying the agreed price. For a Web service, encounter data includes an identity and the available actions are the following: returning the correct result of the invocation in reasonable time, returning incorrect results or delaying the reply. Context data for a Web service may include parameters of the Web service that could influence running times.

3.2.3 Results: Trust, Risk, Credibility and Others

TM service returns values of trust, risk, credibility or other information, depending on the type of invoked service. Note that the library is capable of incorporating various definitions of these concepts [143]. For example, we can adopt the definition of *trust as a tolerance of risk*. This definition emphasizes that trust and risk can be values expressed on the same scale, thus enabling a direct comparison (see section 2.2).

For the Internet auction, the TM service can return the reputation represented on a simple ordinal scale. In the case of the Web service, trust may be not represented at all. The TM service could return information about the policies that have been satisfied by available information. In addition, the TM service can run the risk that a service takes an excessively long time to return a result.

3.2.4 Feedback by Universal Proofs

After the encounter, the agent passes feedback information to the TM system. To represent this information, the library uses the concept of a Proof, a universal representation of an encounter.

In the Internet auction, a proof is a history-based report. In the Web service, a proof, called a "security token", delegates trust obtained in previous encounters with a trust management authority. Proofs can be prior even trust assumptions. Proofs can also be added at any other moment. In the Web service example, proofs could be presented by the agent at the beginning of the scenario, possibly following a trust negotiation procedure.

3.2.5 Architecture of TM Services

A library of universal trust management services must be designed so that many different applications could use common primitives and data. It should be able to incorporate diverse methods, various algorithms and protocols of trust management. The challenge in the design of a library of universal TM services is therefore the *discovery of a common basis for the largest possible set of TM methods*.

This section and a class diagram in Figure 3.2 describes the *basic building blocks required to design various trust management services*. We start with an

Encounter and a Proof, already introduced in the previous section. An Encounter includes information about Context, about the participating Agents, and about the available Actions and their outcomes. A set of template encounter definitions can be created in the TM library when a service is developed for a specific application. Though, the instance of an Encounter will be received during service invocation. In the previous section, two examples of encounters were described. In the Internet auction, an Encounter represents a transaction between a buyer and a seller. In the Web service, an Encounter is a Web service invocation or an attempt to obtain "security tokens" from a TM authority.

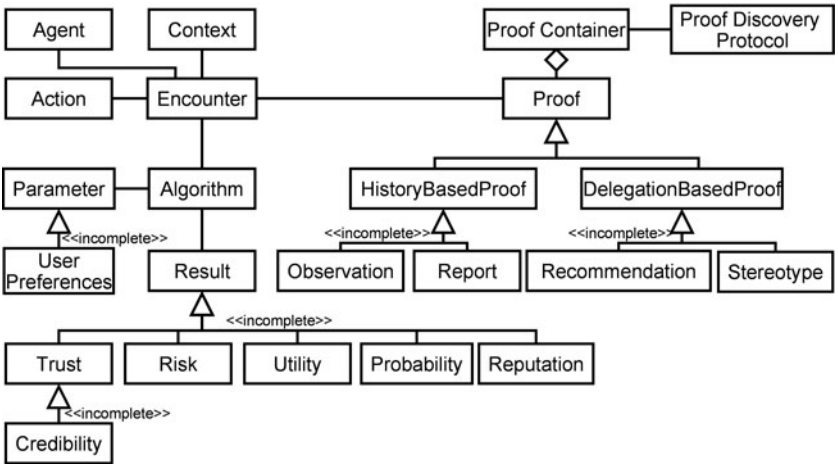


Fig. 3.2 Class Diagram of a TM Library

Note that an Encounter can also be used to model an interaction between two agents who exchange Proofs. The treatment of exchange of Proofs (for example, during reporting in an auction service, during trust negotiation, or during gossiping of opinions in a P2P application) as an Encounter emphasizes that TM methods can be used to decide whether to trust the received Proofs. This form of trust is sometimes referred to as credibility [143], which is modeled in Figure 3.2 as a class that inherits from Trust.

A Proof represents any information that can be used by diverse TM methods to compute trust. A Proof can store data about past encounters of the agent running the library, but also Recommendations or Reports from other agents. For instance, reputation systems use mainly propagated information concerning the history of encounters, modeled as Reports. Observations refer to encounter history, as well, but are obtained first-hand by the observing agent - such a possibility exists, for example, on Wikipedia which displays history of entry modifications. On the other hand, TM methods used in Web

Services use proofs that delegate trust, modeled here as Recommendations. It is hard to list or foresee all kinds of proofs that will be used in a TM service library; however, the kinds planned so far should be sufficient to support many diverse TM methods. Proofs are kept in a Proof Container that is under the direct control of the agent running the TM service. However, when additional Proofs are required that are not available, the Proof Container can use the Proof Discovery Protocol to search for new proofs or request them in Encounters from other agents using trust negotiation. Various TM methods use different kinds of information, such as Trust, Reputation, or Risk. The algorithms that actually calculate this information are all modeled as TMAlgorithms. The arguments of TMAlgorithms are usually Proofs, but TMAlgorithms can also use different Parameters such as User Preferences. For example, consider Risk, which could depend on Risk Aversion as a Parameter.

A universal TM system should be able to integrate Proofs of different kinds. Our approach enables the construction of such a system. Also, the distinction between Reports and Observations on the one hand, and Recommendations on the other, makes it possible to further understand the relation between reputation and computational trust.

Consider a situation where the TM system only has Observations issued by agent A about agent B, all in the same encounter context. This information can be used to calculate the reputation of agent B as perceived by agent A. In the terminology introduced in section 2.2, this is direct, local reputation. (If agent A has no Observations about B, direct, local reputation is undefined; however, if A has Reports about B from C, then it may be able to calculate an indirect, local reputation of B.) If agent A has calculated a direct reputation of B, then he may issue a Recommendation about B. Recommendations can be thought of as information about computational trust. Computational trust propagation algorithms that will be discussed further in this chapter can be used to calculate computational trust between agents that do not have a history of direct interactions.

The question remains, then, what is the difference between a TM system that uses only Recommendations to express computational trust, and a TM system that uses Observations to calculate direct reputation that is the basis of computational trust? For comparison of these two systems, we have to assume that the Recommendations and Observations are all in the same encounter context. One of the differences is the consideration of time. The behavior of agents can change over time, and it is possible for adversaries to strategically modify their behavior (for example, to earn high trust, and then abuse this trust). Thus, a sequence of observations made by A about B should be indexed by time, and can be denoted by O_t^{AB}. There can be many varied methods used to calculate direct reputation in a way that considers time (an overview of these methods will be given lather). For example, it is possible to disregard time and just calculate an average, or simply the most

recent Observation. It is also possible to use weights that increase for more recent observations (linearly, quadratically or exponentially).

Another difference lies in the possibility of error in the evaluation of an agent. If the Observations made by A can be impacted by some noise (for example, in an Internet auction, the performance of a seller is also dependent on postal delivery, and the quality of a Web service service is dependent on transient network congestion), then Observations should be made repeatedly in order to better approximate the reliability, quality, or fairness of an agent. In such a situation, the distribution of all available Observations should be taken into account when calculating a direct reputation.

3.2.5.1 Activities of a TM Service Library

This section and Figure 3.3 show *how the classes introduced in the previous section interact with each other.*

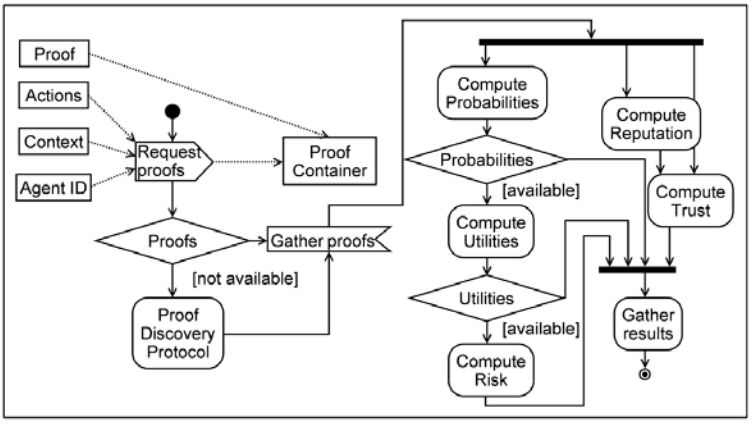


Fig. 3.3 Activity diagram of a TM Services Library

When a TM service is invoked, it formulates a query for proofs. The query is passed to the Proof Container and can result in the invocation of the Proof Discovery Protocol. Note that an Encounter can be the result of receiving a Proof from another agent; this Proof is forwarded to the Proof Container as shown in the figure. The next activity of the service library is the gathering of proofs, which can then - depending on their type - be used to compute various kinds of information, using functions stored in the library. The diagram shows the computation of Reputation, Trust and Risk, emphasizing that information about action probabilities and action utilities is necessary to calculate Risk.

3.2.5.2 Centralized or Distributed Trust Management?

In our opinion, a universal TM library should support both centralized and distributed operation. In the approach presented, Proofs are received during Encounters and stored in the Proof Container that is under the direct control of the agent running the TM service library. This does not exclude the possibility of other agents being present who also have their own Proof Containers. However, agents can also use TM services provided by one, trusted agent who centrally gathers all Proofs. If a more distributed approach is used, agents can exchange Reports or issue Recommendations about other agents, using for example a P2P gossiping protocol or a structured P2P overlay, which can be instances of the Proof Discovery Protocol. Any algorithm to be discussed further can be executed using all available information in the local Proof Container; the question is whether the results of the algorithm will be useful if the information in the locally available Proofs are incomplete.

Another approach to distributed TM requires the use of iterative or recursive algorithms that are inherently distributed and require communication. This approach is not chosen in the proposed TM library for several reasons. Firstly, the convergence of such an algorithm is always dependent on the dynamics of the system. If new Proofs are made available by agents in the TM system all the time, then such an algorithm may never converge. Secondly, the communication cost of such an algorithm may be large. For these reasons, in the anticipated design of the universal TM system we propose to decouple the discovery of proofs from the computation of trust or reputation through the use of the Proof Discovery Protocol that is able to operate independently of the TM system's computations.

3.3 Evaluation of TM Systems

While many of the TM methods presented in literature have been evaluated by their authors, there is still no established and widely accepted method of TM evaluation - much less any form of a benchmark for methods. In the uTrust project, the development of such evaluation methods and benchmarks is planned. Here, some initial ideas are presented.

3.3.1 Effectiveness of Trust Management

A Trust Management service that consistently overvalues, or undervalues trust (or risk) certainly decreases users' performance. It is necessary, therefore, to consider Trust Management service correctness. This could be validated by using special scenarios that can be treated analytically in order to calculate correct values of trust and risk. Next, various Trust Management algorithms could be simulated using the considered scenario, in order to see how close the resulting values are to the correct values of trust and risk.

3.3.2 Adversary Models and TM Benchmarks

One of the most important aspects of Trust Management evaluation is the resistance of TM methods to adversaries. Since many Trust Management methods - especially all reputation systems - are based on a majority principle, they are also vulnerable to colluding adversaries or to strategies such as discrimination [34]. Here, we discuss the characteristics of adversaries against which the services library should be evaluated. Adversary models depend on the *model of the environment* in which Trust Management service works. Such a model states what basic security services are available. For example, a strong, certificate-based authentication of all agents makes many schemes of adversary behavior impossible. However, in other models of environments, only weak, pseudonymous authentication can be available. The environment model should also specify other relevant aspects of information security, for instance whether the communication primitives are vulnerable to eavesdropping, modification, or man-in-the-middle attacks. After determining an environment model, different adversary models can be chosen. An adversary can be described by the following characteristics:

Adversary knowledge. An adversary's knowledge about the TM system can vary. An *omniscient adversary* knows all the used Trust Management algorithms and attempts to exploit their weaknesses; an ordinary adversary uses rather simple, sub-optimal strategies. Even if an adversary knows the used Trust Management algorithms, he may or may not have access to the complete information used by the TM system. Adversaries can thus have *global or partial knowledge* of the information used by the Trust Management system.

Adversary goals. An adversary could be *selfish*, pursuing only her individual gain, for example a maximization of her own utility, performance, or reputation. On the other hand, an adversary can also be *malicious*: he can aim to damage the ODS, or to decrease the utility, performance or reputation of other agents. In the latter case, when an adversary is malicious towards particular agents but benign or selfish towards others, we speak of a *discriminating adversary.* Adversaries may also be capable of *adapting* their strategy.

Adversary collaboration degree. An adversary can act *autonomously*, or can be capable of *collaboration* with other adversaries. The kinds of collaboration could be limited (for example, the number of collaborating agents could be limited).

Adversary resources. Adversaries could have varying amounts of resources of various types: computational resources, communication resources, or even control over bots that can act as fake agents. Adversary resources would be especially important in comparison with the resources available to other agents who use the TM system.

Adversary complexity. Adversaries can have varying degrees of complexity. We can consider *strategic and intelligent* adversaries, who are capable of unlimited computation and unconstrained reasoning. Other adversaries may be *heuristic* or *reactive*, capable of only limited computation and of applying rules that react to events, rather than of strategic reasoning. Adversaries may also employ a variety of algorithms or strategies that can vary in complexity.

3.3.2.1 Adversary Models for P2P Trust Management

Several adversaries have been proposed for P2P trust management. Kamvar [69] describes a number of threat models: an individual malicious peer that always provides an inauthentic file; malicious collectives that assign maximum trust values to other malicious peers, and malicious spies who provide good files but assign maximum trust values to malicious peers. Malicious peers can provide inauthentic files with a given probability.

[32] reviews several attacks on P2P networks, such as the self replication attack (an attack that relies on altering data) and man-in-the-middle attacks. They describe the *Mandragore Gnutella* worm that responds to all queries and sends a copy of itself instead of the requested data. Man-in-the-middle attacks can be executed by modifying the IP and port of the QueryHit message and sending corrupted data. Moreover, Damiani et al. introduce attacks on reputation systems called *pseudospoofing* that is a type of whitewashing attack, and the *ID stealth* attack that uses the Sybil method to multiply peer IDs during query and vote phase. Such an attack aims to boost the reputation of the attacker (shilling attack).

PeerTrust [184] list several adversary models such as modifying information (providing fake resources), man in the middle attacks, or compromising peer settings (allowing others to share specific files like trojans or viruses). In addition to PeerTrust, Srivatsa et al. [155] consider malicious peers referred to as "strategic" who alter their behavior based on their own reputation value. Also, they describe shilling attacks to boost malicious peers' reputation.

Abrams et al. in the *cycling partitioning* [2] system assume the existence of two types of adversaries: malicious and selfish. Selfish peers wish only to maximize their trust score. Malicious peers desire to minimize the number of authentic downloads in the network and may collaborate to do so.

The dropping of negative proofs has been described in R-CHAIN[99]. Malicious nodes would like to keep only favorable *transaction records* and drop unfavorable records.

[88] assume that malicious nodes can disseminate arbitrary trust information. All malicious peers in NICE form a cooperating clique. Each malicious peer always reports implicit trust (maximum value) for every other malicious peer and has a 50% chance of truthful reporting for normal peers.

Adversary strategies vary from simple to complex ones and can pose varying levels of threats. We shall use the adversary characteristics developed in section 3.3.2 to distinguish between various types of adversaries in P2P trust management systems.

Knowledge. Adversaries may differ in their knowledge about the system. We can distinguish between adversaries who rely only on their local knowledge; more advanced entities who have knowledge about their neighborhood within the radius k, and adversaries with full system information. To become an adversary with global knowledge, a malicious peer should gather information about the P2P system, for example using a crawler. The adversary can gather knowledge about the ranking of peers, network interconnections or data distribution.

Goals. Adversaries may have a variety of objectives. There could be *selfish* peers [2, 126, 78, 149], whose purpose is simply to boost their reputation, performance or utility. Such peers can form collectives [55, 69, 121] to obtain their goal.

Another type of adversaries are *malicious*. Their aim is to destroy the P2P system itself (or a part of it) by damaging the query or/and response messages [2], sending corrupted data [69, 32, 148, 184] or simply by making peers inoperative [148, 184] or compromising peers [184]. Adversaries can also attack the Trust Management system itself by executing attacks such as altering feedback information [88, 69, 99], man in the middle attacks [32, 184] or whitewashing attacks [32, 148].

Some *adaptive* adversaries, described in [155], can change their goals depending on their own reputation value.

Collaboration degree. The goals of an adversary affects his collaboration degree. Some goals cannot be achieved by a single adversary, due to their difficulty or to the possibility of detection. Adversaries can act *autonomously* (e.g. selfish peers), or *in collaboration* with other adversaries. Collaboration may be occasional, or peers can form a *collective* (*clique*) [69, 88, 148].

Complexity. Like in the previous aspect, the complexity of an adversary depends on his goals. Simple attacks may be performed by *dropping messages* [99], *tampering with data* [69, 32, 148, 184, 2] or *altering feedback* [88, 69, 99].

Complex attacks can be a combination of simpler attacks. For example, the *denial-of-service* attack [148] can be executed by peer collectives. A DoS attack can affect the P2P overlay itself, for example crash some highly connected peers to handicap or even partition the network. This can be exploited by peers who want to modify routing or spoof the identity of other peers.

Adversaries can also perform the *Sybil attack*[69, 32] to create a number of clones of themselves using different IDs. This attack can be combined with *whitewashing* to boost the reputation of a peer.

3.3.3 Scalability and Efficiency

3.3.3.1 Evaluation of Computational Cost

A neglected, yet important aspect of Trust Management evaluation is the investigation of TM method performance. Many TM methods, in particular cryptographic and Bayesian TM, are rather complex, resulting in a non-negligible computational cost. In the context of distributed systems, some TM methods can also incur communication overheads.

3.4 Authentication Requirements

A trust management system could use many different forms of authentication. At present, many applications in ODS use weak authentication based on nick names and IP addresses (or IDs that are derived from such information). However, it has been shown that such systems are vulnerable to the Sybil attack [42].

Most of the mechanisms discussed in this section would not work if the system were to be compromised using the Sybil attack. An attacker that can control an arbitrary number of clones under different IDs could use these clones to defeat the TM system. One way to prevent the Sybil attack is to use a strong form of authentication, such as one based on public-key cryptography. Public key cryptography may be used in a TM system for authentication and digital signatures of short messages.

In applications of ODS, authentication must be used efficiently. In other words, it should not be necessary to repeatedly authenticate agents. The use of authentication could depend on the application. For instance, if groups of agents interact frequently with each other in an ODS, perhaps only mutual authentication of group members is required. Once all agents in the group are authenticated, they could agree on a common secret (such as a group key) that will be used to identify group members, using a method such as the Secure Group Layer (SGL) [15]. Alternatively, these users could exchange public keys.

Let us briefly discuss here how such an authentication could be implemented. A well-known solution is the PKI infrastructure. This solution also has the advantage of direct availability and possibility of implementation. However, developers of P2P games may be concerned about the single point of failure, lack of anonymity, or insufficient security of PKI [22]. The use of global PKI may be unnecessary, since certificates can be issued for example by a bootstrap server or coordinator. On the other hand, such a design assumes the

existence of a trust relation between the agents in the ODS and the identity responsible for issuing certificates. There may exist solutions better suited to the needs of ODS, such as the Web-of-Trust model, or solutions based on trust graphs. Also, we recognize that the use of local names as proposed in Simple Public Key Infrastructure (SPKI) [154] is more suited to ODS applications and should be also more scalable. Another solution that is well suited to the P2P model is the use of threshold cryptography for distributed PKI [6].

Another, promising direction for protecting against Sybil attack is the use of a social network. Mechanisms like SybilGuard [42] are based on the assumption that malicious peers can create only a few links to honest users even if they possess many fake identities. This assumption can be satisfied by establishing secure edges between user pairs by using shared/secret key, exchanged among the participants using an out-of-band mechanism. Another interesting possibility is to let the trust management system itself control the edges (the social network used by SybilGuard is then the trust network). This approach ought to assure that adversaries that can be recognized by the TM system receive few and weak edges, which renders them unable to attack a mechanism such as SybilGuard.

3.5 Computational Trust

As defined in section 2.1, computational trust is a representation of human trust used in trust management systems. Similarly to human trust, computational trust is usually modeled as a relation between agents, although some trust management systems use a representation of trustworthiness (often referred to as *global trust*) that is the property of an agent and not a relation between agents. In this section, we shall refer to computational trust only in the sense of a representation of human trust relation (in this meaning, the term *local trust* is sometimes used in the literature). We shall use the following notation: $TR =< V, T >$, where V is the set of agents and T is the set of trust relations between the agents, denotes the computational (trust relation). Each edge $e \in T$ is associated with a strength s_e that is a specific computational trust value. The values of s_e can be obtained from Recommendations or from direct reputation that is calculated from Observations. We shall use the notation s_{AB} if $e = (A, B)$ is an edge from node A to node B. Computational representations used for s_e differ in different approaches to trust and will be described further in this section. The computational distrust relation is denoted by $DTR =< V, DT >$, where V is the same set of agents as in TR and DT is the set of distrust relations among the agents. Similarly, every edge $de \in DT$ is associated with a strength s_{de}. We shall use the same notation for the strength of the trust and distrust relations.

3.5.1 Simple Computational Trust Models

Most Trust Management systems use simple computational representations of trust. Internet auctions, for example, use a three-valued discrete scale of "negative", "neutral" and "positive" (with the exception of the new system used by e-Bay). The epinions recommendation system [109] also uses a three-valued scale. The FilmTrust [54] recommendation system uses a 10-valued discrete scale. More theoretical reputation systems often use a continuous scale from $[0, 1$ or $[-1, 1]$ [118, 108]. A more complex system has been proposed by Josang [65] and uses a two-dimensional scale with uncertainty representation. However, Josang's system also uses a simple scale for the actual representation of trust expressed by users.

The questions considered in this section are: what is the appropriate system for computational representation of human trust? Are the simpler systems used so-far sufficient for this purpose? This question has been frequently considered in the literature on trust management; yet, previous research has lacked a way of evaluating the simpler trust representations, and has therefore usually opted for simplicity. Another reason for this choice is that the consequences of introducing a new computational trust representation for the TM system are not simple. A simple representation makes it easier to develop meaningful algorithms for processing trust. One of the most important types of algorithm used by a TM system is the trust propagation algorithm. A simple representation of trust makes it easy to compute a propagated trust value (for example, using multiplicative trust propagation). Therefore, in order to answer our question constructively, we must not only propose a suitable trust representation, but also to define how the new computational trust can be processed by the TM system.

To tackle this issue, it is necessary to use information on how users express trust. The difficulty here is that most available datasets are obtained from TM systems that use a simpler computational trust representation. However, these systems (probably in order to account for the limitations introduced by the simplistic computational trust representation) also allow users to add a textual comment. Our approach to the evaluation of how humans express trust will be based on a large dataset of comments from an Internet auction platform. In this dataset, we shall investigate the following questions: do users systematically try to add more information than is allowed by the computational trust scale used by the reputation system? What kind of information is being added by users? How are the quantitative reports (using the current computational trust scale) related to the additional information in textual comments? In order to answer these and related questions, we shall use Natural Language analysis of the textual comments that extracts their emotional content, as well as a classification of comments that reveals an implicit, more complex scale of trust valuations.

Based on the analysis of the dataset, we shall attempt to propose a trust representation system that is adapted to the implicit user requirements

revealed by our analysis. The proposed system is based on the work of Sabater-Mir [146]. It is also in some respects similar to eBay's new Detailed Seller Rating system[2].

Therefore, we shall review the DSR system and discuss some of its deficiencies. We shall also introduce operators that allow to process the new computational trust. In this area, we go beyond the work of Sabater-Mir, introducing new operators and showing how the proposed computational trust can be used in a real, general trust propagation algorithm. We are able to show that our proposed computational trust representation can be used in various types of propagation, while at the same time it is much better adapted to an expression of human trust, based on our observations of the Internet auction trace.

3.5.2 Empirical Investigation of Human Trust Expression

Reputation systems in Internet auctions are perhaps the most widely used form of a TM system today. For that reason, they are an attractive choice when we consider the question of adequacy of computational trust representation. An attempt to answer this question could choose one of two approaches: directly asking users about their opinion on the trust representation, or analyzing a dataset that records actual usage of the reputation system. The first approach has the following drawbacks: it would have to be limited in the number of received opinions (as with any survey-based approach); the received opinions may be biased based on the actual user experiences with the auction system; and, it would be hard to propose constructive improvements to the trust representation system based on the received responses (we could ask users to compare various versions of trust representation or ask about specific improvements, but their answers would not necessarily be conclusive). Also, a survey-based approach would suffer with the problem of choosing a representative set of responses, and would rely entirely on declarative data, which may be imperfect due to human memory or bias.

For these reasons, we have chosen the second approach. Based on a large trace from an Internet auction site, we shall analyze the usage patterns of the reputation system, with particular emphasis on behavior that indicates whether a user is trying to circumvent the limitations imposed by the computational trust representation. This involves searching for responses to the following questions:

- do users systematically try to add more information than they can express on the scale used by the reputation system?
- if they do, what kind of information is being added?

[2] http://pages.ebay.co.uk/services/buyandsell/powerseller/criteria.html

- how is this information related to the values expressed on the scale of reports?
- do users try to extend or reduce the available scale of reports in their comments?

Our approach is based on textual comments added to reports sent by users of the Internet auction system. These reports represent the users' valuations, typically on a three valued scale of "negative", "neutral" and "positive". It is noteworthy that reputation systems used in Internet auctions have evolved and recently, eBay has proposed a new scale referred to as "Detailed Seller Rating" (DSR). In the textual comments, users contain a wealth of information about each other. However, it is interesting that users usually add information only for negative or neutral comments. For positive comments, the added information is negligible.

In the textual comments, we shall search for the following kinds of additional information: that which expresses the emotional attitude of the reporting user. Here, we shall be especially interested in whether users try to amplify the meaning of the report by a strongly emotional textual comment. Emotionality of comments can also be used in order to judge whether users agree with the proposed scale of feedback. If they do, then the comments for "neutral" feedbacks should be emotionally more positive than "negative" feedbacks, and less positive than "positive" feedbacks. If users do not exhibit this kind of behavior, it means that they are not using the feedbacks according to the initially designed scale. This could be due to the fact that users evaluate feedback differently: not on a simple scale that constitutes a total order. We have found evidence that this is the case.

Another interesting type of additional information comes from an attempt to describe the situation that has occurred during the auction or transaction. If the added descriptions form a systematic set of categories (a taxonomy), then the users are actually modifying the scale used by the reports using a systematic set of descriptions. Once again, we find evidence that this is the case. The set of categories used depends on the type of report (negative or neutral), and is absent for positive reports.

Once again, the question of feedback valuation can be reconsidered if a taxonomy of non positive feedback categories exists. It is then possible to pose the question: do users attach the same importance to all categories of feedback? Or rather, does there exist some kind of relation that allows us to compare the importance of various feedback categories? Studying the emotional content of comments is one way of answering this question. However, we have also considered the balance of negative and neutral feedback for each category and have carried out an opinion poll among buyers and sellers on a Polish Internet auction site. In the poll, we asked auction users to compare the importance of categories. We found evidence that there exists a relation that allows us to compare categories according to their importance.

3.5.2.1 The Used Internet Auction Trace

The dataset was acquired from *www.allegro.pl* which is the leading Eastern European online auction provider. In this service, each auction has an explicit deadline and all current bids are exposed to all participants. Moreover, all information about all participants is accessible. In most auctions the bidders can specify a maximum price that they want to pay for an item and the proxy bid system automatically raises the bid, using only as much of the bid as is necessary to maintain the top position. Bidders can also increase their maximum price at any moment. When the auction terminates, the bidder with the highest bid wins. There are also multi-item (*Buy now!*)-type auctions in which sellers can sell more than one item (and hence there is more than one winner). In such an auction, every bid is the winning bid.

We have selected a subset of 9500 sellers and buyers, and their $285K$ auctions listed in $16K$ categories. Our research was performed on a subset of $1700K$ cases of positive and non-positive ($15K$) feedback. The unequal amount of auctions and feedback items is caused by the existence of multi-item auctions.

3.5.2.2 Internet Auction Comment Classification

Existing reputation systems do not distinguish between different kinds of negative or neutral user feedback. As long as we treat every negative feedback equally, we cannot distinguish purposeful behavior from an accidental one. For example, there is a great difference between sending the wrong color or size of a T-shirt and not sending it at all.

In our previous [73], we mined information from research users' comments using two independent classification rules for the buyers and for the sellers. We have partitioned all negative and neutral feedback into the detailed types of the complaint taxonomy, using regular expressions. Each complaint type has its own meaning and also a unique set of regular expression patterns. We created tree structures of complaints against buyers and sellers similar to [55].

A feedback entry could be assigned to more than one pattern from different types. Our regular expression tool has matched 68% of negative comments (for the seller and the buyer equally), 54% of neutral comments for the seller and 35% of neutral comments for the buyer. Figure 3.4 shows the taxonomy of complaints against sellers. The most detailed classes of this taxonomy are sometimes too small, so we have opted to present our analysis for a higher level of the taxonomy tree, marked as "General" on the figure.

We observed two types of harmful activity reported by auction users: user behavior related and item related. The first group includes the following user behavior:

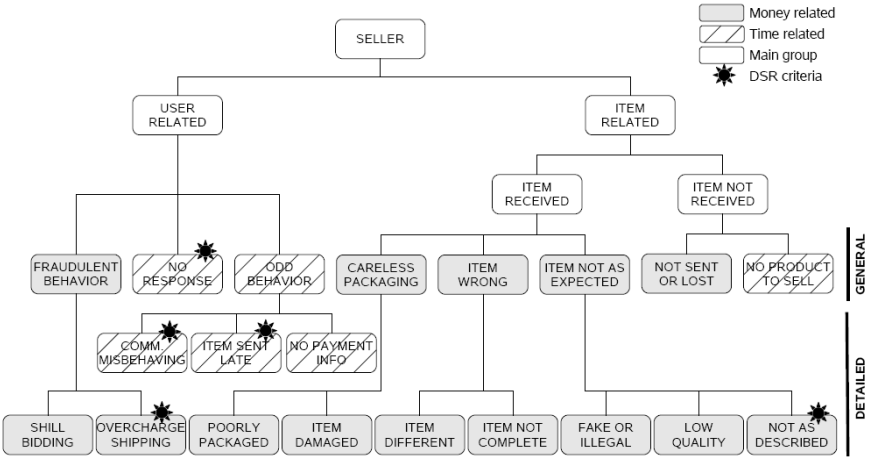


Fig. 3.4 Taxonomy of complaints against sellers

- **No response.** Communication with the user after the auction was impossible. The user did not answer phone calls and did not respond to e-mails. The meaning of this behavior is equal for sellers and buyers.
- **Odd behavior.** The user behaved in a completely unpredictable manner, communication with the seller was possible but handicapped. If user has the seller role, she has sent the item with a delay or has not defined the payment method and shipping price. If the user is a buyer, she seemed not to follow the auction rules, or did not read the information provided by the seller. Sometimes a buyer even tries to force the seller to choose a particular payment method.
- **Delivery not accepted.** The buyer did not accept the delivery which should be paid for by cash on delivery. The seller must pay the round trip shipping charges, which is sometimes a significant amount of money. This is the only type of complaint against the buyer related to loss of money.
- **No intention to buy.** The buyer did not pay for the item, and did not inform the seller about her plans. Sellers call such behavior childish or bidding for fun.
- **Reneged on buying.** The buyer contacts the seller and declares that she will not buy the item. From the seller side, the category 'No product to sell' is very similar to this behavior.
- **Overcharged shipping or shill bidding.** We consider only explicitly formulated accusations concerning shill bidding or shipping overcharge, not those computed from historical auction data. Such behavior is related only to the seller.

The second group of complaints is related strictly to the item (and thereby to the seller) and consists of:

- **Item not sent or lost.** The item was not sent to the recipient. Sometimes the seller argues that the item was lost by the courier or post office.
- **No product to sell.** The seller declares that the item was already sold to another buyer, or the item is no longer on sale. In this case the item is not sent to the buyer.
- **Careless Packing.** The seller did not take care about the packaging of the items. This type also includes the situation when the received item was damaged. It is not possible to verify whether the seller has sent a damaged item or the item has been destroyed during shipment.
- **Wrong item.** The seller made a mistake and sent a wrong item (wrong color or type) or the received item was not complete.
- **Item not as expected.** The item seems to be illegal goods (a fake, or a pirate copy of software) or just does not satisfy the buyer.

The results of our complaint classification show that users use a rich and complex system for expressing the typical types of behavior in Internet auctions. This is in contrast to the simplistic representation of comments as negative, neutral or positive. Negative and neutral comments could benefit from a much more detailed description of reported behavior.

An important conclusion from the observed classification of complaints is that almost all of these categories are related to *procedural fairness criteria of Internet auctions* (with the exception of the "Odd behavior" category). This observation points out that users of Internet auctions have created an implicit code of behavior (that is supported by an explicit statement of rules by the Internet auction platform) and are using the rules of that code in the comments posted in feedbacks. This conclusion gives empirical support to the hypothesis that the concepts of fairness and trust are related, as proposed in the trust definition given in section 2.2.1.1.

3.5.2.3 Emotional Variation of Internet Auction Comments

Automatic Sentiment Extraction

For the sentiment analysis task we used a modified version of Sentipejd [20] - a hybrid of lexeme category analysis with a shallow parsing engine. At the basic level, Sentipejd checks for the presence of a specific category of lexemes. Such an abstraction originates or in content analysis systems, most notably the classic General Inquirer [132]. Lexical categories used in this work include two sets of words (dictionaries): 1580 positive and 1870 negative ones, created by Zetema[3]. Because comment texts are typed in a careless manner, very often completely without diactrits, lexeme recognition was extended with a diacrit guesser. Recognized sentiment lexemes, along with morphosyntactic tags, are analyzed with Spejd - a tool for simultaneous morphosyntactic disambiguation and shallow

[3] www.zetema.pl

parsing [19], with a number of rules crafted to recognize multiword opinion patterns and apply sentiment modifying operations.

The Spejd formalism is a cascade of regular grammars. Unlike in the case of other shallow parsing formalisms, the rules of grammar allow for explicit morphosyntactic disambiguation, independently or in connection with structure building statements, which facilitates the task of the shallow parsing of ambiguous and/or erroneous input.

For the purpose of sentiment analysis we extended the default Spejd's morphosyntactic tagset with a sentiment category expressing properties of positive or negative sentiment. This hybrid approach is called Sentipejd [20]. Sentiment rules included (but were not limited to): affirmation, negation, nullification, limitation and negative modification.

Table 3.1 Emotion intensity and comments' length in categories

Feedback category	Emotion	Avg. comment's length
POS	2.75	89.38
NEU	0.37	161.66
NEG	−1.08	177.00

The most commonly used reputation systems embedded in online auction website allow us to evaluate transaction results not only by selecting a predefined category from a list but also by leaving longer or shorter comments. When a given transaction is completed, every eBay/Allegro user can evaluate his or her partner by choosing either a positive, neutral, or negative mark. General, common-sense driven understanding of these three textual labels as three separate categories which represent three grades of transaction outcome evaluation, does not stand up to scrutiny of data from the real system. 99% of transactions are followed by comments labeled "positive" and only 0.6% and 0.4%, respectively, neutral and negative. The overwhelming domination of positive evaluation indicates that users, both buyers and sellers, award every non-fraudulent outcome of transaction with positive comment[4].

The average comment's lengths for separate feedback categories are presented in tab. 3.5.2.3. It is easily noticeable that as far as comment's length is concerned the difference between positive and neutral comments is much bigger than between neutral and negative. Neutral comments are long (in comparison to positive). More characters are probably needed for expressing a user's attitude (evaluation) toward an unexpected transaction's outcome.

As far as emotional intensity is concerned the neutral comments are also closer to negative than positive. The value of emotion index ranging above zero should not be mistakenly understood as an expression of satisfaction.

[4] In some cases positive comment is coerced by either unjustified reciprocal evaluation or legal threat.

Because of the properties of emotional measurement selected for this research a meaningful interpretation of results can only be performed based on relative values. Yet another approach to in-depth study of emotional distance between categories was applied. Two separated, balanced subsets of comments were created:

- Set I (NEG; NEU) - contains 1454 comments of which 727 are negative and 727 positive,
- Set II (NEU; POS) - contains 1454 comments of which 727 are neutral and 727 positive.

Every set of comments has been partitioned on testing and training set (30% and 70% cases respectively). For both sets of comments three different classification algorithms were used: neural network, support vector machine and decision trees (CHAID algorithm). As a target variable the label given by the comment's author (negative, neutral or positive) was selected and the emotional intensity as input variables.

Table 3.2 Classification accuracy for different algorithms and testing subsets

	Neural Network	Support Vector Machine	Decision Trees (CHAID)
NEG and NEU (set I)	65, 16%	65, 80%	64, 11%
NEU and POS (set II)	65, 16%	65, 80%	64, 11%

To confirm the hypothesis that neutral comments are more similar to negative than positive the quality of classification for two sets described in previous paragraph were compared. Independent of used algorithm obtained results show that it's easier to distinguish between neutral and positive comments than neutral and negative. Thus, the claim that in a real reputation system embedded in an auction house only positive and non-positive comments exist. In most applications negative and neutral comments can be interpreted in the same way - as an expression of dissatisfaction - and the label should not be explicitly treated as a scale of the experiences.

The conclusion from the analysis of emotional content of comments is that Internet auction users do not use the proposed three-valued scale of comments in an anticipated manner. Rather, they distinguish mainly between positive and nonpositive feedback. On the other hand, the difference between these two kinds of feedback is very large and even amplified by the emotional content of comments. These findings seem to question the validity of using such a simple scale. One possible conclusion is that it would suffice to use a binary scale, as proposed by many trust management systems. However, as our advanced analysis shows, this situation is really more complex, because users do not evaluate feedback on a scale that is a total order and instead use many different criteria when considering nonpositive feedback.

3.5.2.4 Harmfulness of Unfair Behavior

A simple computational method can be used to rate the types of complaints along with their harmfulness. This method is based on the percentages of negative and neutral comments in each complaint category. The balance between the negative and neutral comments serves as a measure of the harmfulness of this type of complaint.

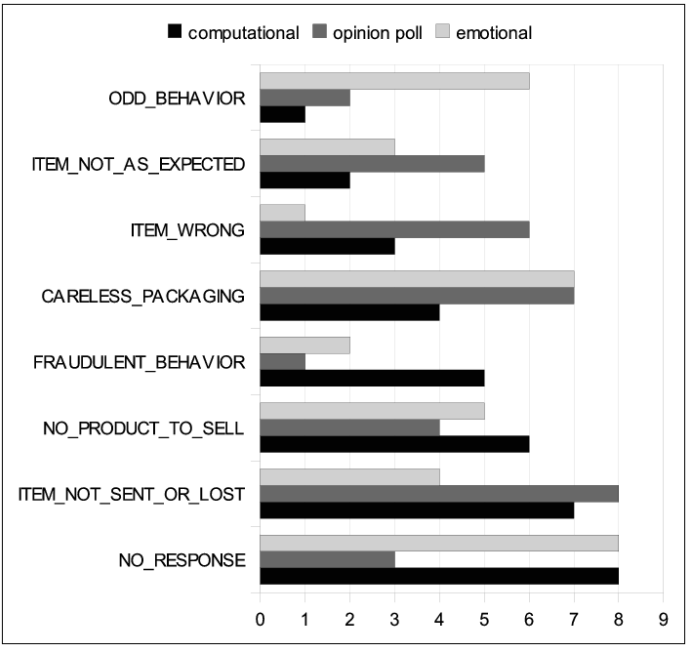


Fig. 3.5 Harmfulness grading of sellers' activities

In order to verify the correctness of this method, we also conducted an opinion poll among real Internet auction users. We received 208 responses from Internet auction users (between the ages of 19 and 59). 148 respondents declared that they had sold goods on auction systems, and 193 of them declared that they had bought goods on auctions. This implies that a large number of the respondents have been both buyers and sellers. Respondents were asked about their subjective opinion on the *harmfulness* of each complaint category.

In addition we included an evaluation of the emotional content of feedback description. All values for the computational and emotional methods were generated from nonpositive (negative or neutral) feedback. We have juxtaposed the results from all three methods in Figure 3.1 and Figure 3.2 (for sellers and buyers respectively). The Figure shows the harmfulness of each complaint category as calculated by all three methods. The length of the bar

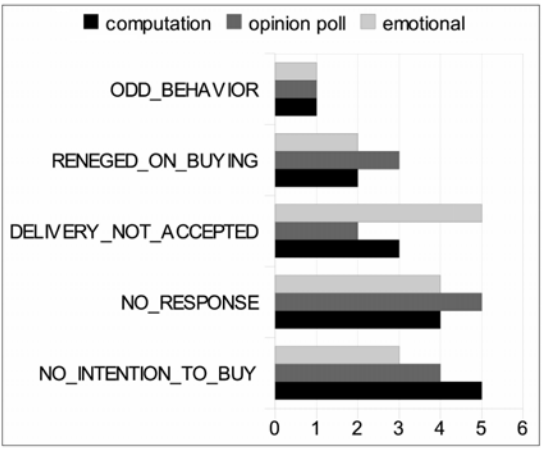


Fig. 3.6 Harmfulness grading of buyers' activities

indicates the position of the particular category in a ranking created by the particular evaluation method.

The results obtained from this opinion poll demonstrate that the more frequently used types of complaint are not necessarily the most *harmful* as considered by our respondents. Users seem to be more tolerant to lack of response from the seller or to a situation in which the seller declares after the auction that there is no item to sell. According to our respondents, the most harmful behaviors concern the condition of the item, such as sending a damaged, incomplete or different item.

From the emotional point of view, buyers are more irritated when there is a communication problem with the seller (odd behavior or lack of response). Buyers get more emotional when the seller declares that there is no product to sell, or the product is damaged.

From the seller's point of view, the most harmful behavior is related to the lack of response and lack of payment (no intention to buy). There is a minor disagreement between the computational method and the opinion poll. Odd behavior of the buyer is not a serious problem for the seller as long as the buyer still pays before receiving the item. Results from the emotional method show that the most irritating behaviors are not accepting the delivery and lack of response. Both of these groups are related to the loss of money. When the buyer does not pick up the delivery, the seller must pay the round trip shipping charges, which is sometimes a significant amount of money. Lack of response is connected to the loss of time, which (in case of the e-market) is very closely connected to the loss of money.

Based on the three harmfulness rankings of the report categories, it is possible to create an assignment of the categories to importance classes (the more harmful categories should be considered the more important). Let us assume that there are 5 importance classes, *least important*, *less important*,

important, *more important* and *most important*. The problem of assigning the report categories to importance classes based on the three different rankings is really a problem of merging social valuations. The computational methods of the theory of equitable optimality can be used for such a problem; notably, the OWA and WOWA methods are particularly suitable (see section 2.3.9). Recall from this element of study that the WOWA method uses two sets of weights: the priority (or entitlement) weights that are assigned to individual criteria (or agents' outcomes), and the OWA weights that are assigned to the ordered criteria. Consider a WOWA aggregation of the three rankings that will have the following priority weights: $v_c = 2$, $v_e = 1$, and $v_o = 1$, where v_c is the weight of the computational ranking, v_e is the weight of the emotional ranking and v_o is the weight of the opinion poll. This means that the computational method can be counted double, because it is based on the largest set of declarative responses (the emotional method is computed from the contents of the comments, and is therefore not declarative, and the opinion poll is based on a smaller set of responses). This can be achieved by simply cloning the computational ranking, so as a result we obtain 4 rankings where the computational ranking is repeated. For these 4 rankings, we can specify a set of Ordered Weighted Average weights: $w_4 = 0.5$, $w_3 = 0.25$, $w_2 = 0.25$ and $w_1 = 0$, where w_i is the weight of the ith worst position in the four rankings. This set of weights can be interpreted as follows: we consider only the three best positions in the four rankings, and the first position is twice as important as the second and third. If the best position is from the computational ranking, it will occur twice in this computation because the ranking is repeated.

This method produces a joint ranking for the report categories. The ranking can be partitioned into five classes using threshold values that are selected so that all five importance classes will be used to compare the report categories. Using this method, we obtain the following assignment of categories of reports about sellers (see Table 3.3.).

Table 3.3 Assignment of report categories to importance classes basing on WOWA aggregation of rankings

Report category	**Importance class**
No response	Most important
Item not sent or lost	Most important
Careless packaging	More important
No product to sell	Important
Overpriced shipping	Less important
Item wrong	Less important
Item not as expected	Least important
Odd behavior	Least important

This assignment will be used in the next section in order to design a method of comparing reports that takes their importance into account.

3.5.3 A New Computational Trust Representation for Internet Auctions

The computational trust representation proposed in this paper attempts to take into account information about user behavior gained through the analysis of Internet auction traces. The conclusions from this analysis point towards an emphasis on for improvement of the computational trust representation currently used in Internet auctions:

- the categorization of comments has demonstrated that buyers and sellers on Internet auctions use detailed and meaningful criteria for the evaluation of other agents' behavior,
- emotional contents of comments and the comparison of categories in terms of perceived harmfulness shows that Internet auction users have a preference structure that allows us to compare the importance of the various criteria.

These two observations are the basis of the new proposed computational trust representation based on the work of Yager [186, 187] and resembles the representation used by Sabater-Mir et al. [146]. However, the processing of the representations by the proposed operators is significantly different due to a fundamentally different interpretation of the values. Firstly, the labels used in the representation are interpreted by Yager and Sabater-Mir as exclusive alternatives that are comparable by a total order (even though the representation makes it possible to specify non-zero strengths for various labels. Still, the labels without strengths can only be interpreted as exclusive alternatives). In the representation proposed in this paper, labels are interpreted as criteria for the description of behavior. These criteria are not mutually exclusive. In addition, there exists a preference relation that describes the importance of the criteria to users.

Consequently, Yager proposed a completely different treatment of opinions represented by his system. Firstly, the strengths used by Yager are normalized, so that they add up to 1 for the different labels. Such a normalization makes sense for exclusive alternatives, but not for non-exclusive criteria. Furthermore, Yager's aggregation operator is based on strength multiplication, not on the OWA operator as proposed here. This is motivated by the fact that a "unit" opinion in Yager's system is an opinion that has equal strengths for all labels. Such an opinion should not change the aggregated result. In our system, a unit opinion does not exist, but the OWA weights ensure that opinions which have strengths close to the mean will have a smaller impact on the aggregated proof.

Sabater-Mir additionally introduces a confidence measure (called "strength of belief in the representation of trust about an agent"). This confidence

measure is really a measure of trust in the context of the credibility of opinions received from another agent. The confidence measure is therefore used in a form of transitive trust propagation that combines credibility trust with a trust rating in another context. However, since Sabater-Mir does not explicitly identify contexts, the confidence measure is added to every credibility rating. A composition operator could be used instead that would explicitly calculate transitively propagated trust in the context of opinion credibility. However, in Internet auctions, such transitive propagation is not useful, and therefore we have not included such an operator or a confidence meausre in our system.

Another difference to the work of Sabater-Mir is the introduction of the selection operator. This operator can be used to create rankings of agents or to reduce the amount of noise in the aggregation by selecting the more extreme opinions.

3.5.3.1 Proof Definition

In Internet auctions, proofs are in the form of reports about the behavior of an agent (seller or buyer). The new computational trust representation consists of a definition of proof and of operators for processing proofs in order to propagate computational trust. A proof can be defined as follows:

$$P^{AB} = \{c, \{l_i, s_i\}_{i=1}^{m}\}.$$

where A and B are agents: A is the trustor and B is the trustee; c is the proof context. In Internet auctions, contexts can be the categories of products, the price ranges, and the role of the user (buyer or seller).

l_i is a set of labels that are used to describe the behavior of a user. m is the number of labels. One can think of the labels as criteria used to describe behavior. These criteria can be determined from the comment categories proposed in section 3.5.2.2. In contrast to the work of Sabater-Mir, we propose that the labels are not comparable by a strong order. Rather, there exists a preference relation that represents their "degree of importance" of criteria, as indicated by our analysis. We shall denote this fact by $l_i \prec l_j$, whenever l_j is considered as more important than l_i by the trustor.

Each proof also includes a related strength value for each label, s_i. The strength represents the degree to which the proposed label applies to the trustor. Initial strength values should be on an intuitive scale that is easy to use by buyers and sellers on Internet auctions. For example, an integer scale of $0, 1, ..., 5$ can be used. However, strengths will later be processed and transformed by the trust management system, so that it can be assumed that strengths will be real numbers in the range of $[0, 5]$.

3.5.3.2 Requirements for Proof Processing Operators

The proposed proofs will be processed by a trust management system that should be capable of presenting the best possible trust recommendation about a new agent. For example, consider a buyer who has never before bought from a certain seller on an Internet auction. The trust management system should produce a recommendation for that buyer that is based on available proofs. Current reputation systems on Internet auctions use a simple average of scores given in available reports. A more sophisticated system could also attempt to propagate trust using seller similarity – incorporating a larger amount of proofs about similar sellers. (Transitive trust propagation cannot be applied in Internet auctions because buyer and seller roles are asymmetric, and information about the credibility trust in other buyers' opinions is not available). Another requirement is the ability to produce a ranking of agents. For example, consider a buyer who has searched for products and found a set of sellers that offer similar products in comparable price ranges. The buyer wants to rank these sellers according to their computational trust measures. Once again, in a simple reputation system, these sellers would be sorted using the reputation value.

Following the work of [58], we shall define two operators that are sufficient to implement simple ot sophisticated algorithms that fulfil the requirements described above. These operators are: the *aggregation operator* $\oplus$ and the *selection operator* $\triangleleft$. Both operators are applied to a set of proofs, P_i^{AB}. Before formally introducing these operators, let us discuss what should be their desirable properties.

The aggregation operator should return a single proof that represents a summary of the information available in a set of proofs. Usually it is applied to summarize the proofs available from different agents about a single trustee B or even for a single relation between a trustor A and trustee B. In a simple reputation system, where reports are on a scale of $-1, 0, 1$, the aggregation operator could return a ratio of nonnegative reports $(0, 1)$ to all reports. Using the proposed new computational trust representation, the aggregation operator will have to be more complex. Since the aggregation operator produces a single proof, the job is to create the new strengths for that proof. The first possibility is to calculate average values for each label l_i from the strengths of that label in the aggregated set of proofs. However, as more proofs will be aggregated, this algorithm would converge to the average of strengths used for each label by the entire population of agents. In order to produce a more diversified opinion, it is desirable to allow more extreme opinions to have more impact on the result.

The job of the selection operator is to choose the most relevant proofs from a set of proofs. Usually it is applied to select a single proof from a set of aggregated proofs about various trustees. This operation can be applied iteratively to produce a ranking of trustees. Also, the selection operator can be applied before aggregation, in order to select the more extreme opinions. Such a

selection has been shown to improve the quality of trust recommendation [54]. The reason for this is that excluding less relevant proofs reduces the noise of the trust recommendation. A simple way of judging relevance is based on the strengths of opinions. Proofs with extremely high or low strengths should be preferred. In a simple reputation system, the selection operator could just return reports that have values of 1 and -1, ignoring the neutral values. In our system, the selection operator can return proofs that are weakly Pareto-optimal with respect to scaled strengths (in other words, the selection operator will not include a proof if it has strengths less or equal to another proof in the set, and one inequality is strict: it is weakly Pareto-dominated by another proof), where the scaling function should take into account the mean strength value for each label. Selecting Pareto-optimal proofs will return the most positive opinions; this process can be repeated using negated criteria, so that the most negative opinions will be selected, as well. The number of selected proofs should be a parameter. Last but not least, the selection should take into account the preference relation that establishes the relative importance of labels.

3.5.3.3 Aggregation Operator

The aggregation operator can be defined as follows. Let

$$P^{AB}_{\text{agg}} = \oplus(P^{X_1B}, \ldots, P^{X_nB}),$$

where $P^{AB}_{\text{agg}} = \{c, \{l_i, s_i^{\text{agg}}\}\}$ and $P^{X_jB} = \{c, \{l_i, s_i^j\}\}$ for all j.

The trust management system should record the empirical distribution of strengths used for each label. This empirical distribution can be used to determine the average user valuations for each label, denoted by $\overline{s}_i$. Another important value is the kth q-quantile of the strengths used for each label, denoted by $\hat{s}_i^k$. The value of q is a parameter, for example $q = 5$ (quintiles). The defining property of $\hat{s}_i^k$ is that $Probs_i < \hat{s}_i^k \leq k/q$, where $k \in 1, ..., q-1$. For convenience, we shall denote $\hat{s}_i^0 = 0$ and $\hat{s}_i^q = 5$ (recall that $s_i \in [0, 5]$.

The new strengths of the aggregated proof are given by:

$$s_i^{\text{agg}} = \sum_{k=1}^{q} w_k \text{ average}\left(\{s_i^j : \hat{s}_i^{k-1} \leq s_i^j < \hat{s}_i^k\}\right).$$

This operation is equivalent to the OWA operator [121, 185] on quantiles. The weights w_k are applied to the averages of values in the range of two adjacent quantiles. On table 3.4, the proposed values for these weights are shown. Note that the weights are designed so that they will emphasize the extreme values at the cost of the values closer to the mean. The mean values will therefore have a smaller effect on the aggregated proof.

Table 3.4 OWA weights for quantiles

k	1	2	3	4	5
w_k	0.3	0.175	0.05	0.175	0.3

3.5.3.4 Selection Operator

The aggregation operator can be defined as follows. Let

$$P^{AY_{sel}} = \triangleleft(P^{AY_1}, \ldots, P^{AY_n}).$$

The selection of proofs can be done using a real-valued scaling function that should be maximized. Therefore, $P^{AY_{sel}}$ is determined by choosing the proof P^{AY_k} that maximizes this scaling function, denoted by $\sigma(P^{AY_k})$. This function, on the other hand, will use scaling functions that operate on the strength values for any label. The scaling function on proofs can be defined as follows:

$$\sigma(P^{AY_k}) = min_i\{\sigma_i(s_i^k)\} + \epsilon/m \sum_{i=1}^{m} \sigma_i(s_i^k),$$

where $P^{AY_k} = \{c, \{l_i, s_i^k\}\}$ for all k.

The scaling function of strengths is based on the partial achievement function of the reference-point approach to multi-criteria optimization [180]. The values of this function can be interpreted as satisfaction levels for the strength values. The scaling function is piecewise linear, but changes its slope at two special points (strength values). These are the so-called reservation (s_i^r) and aspiration (s_i^a) points. These points can be determined from the empirical distribution of strength values, as follows: $s_i^r = \overline{s}_i/2$ and $s_i^a = (5 + \overline{s}_i)/2$, for each label l_i.

The scaling function of strengths is given by:

$$\sigma_i(s_i) = \begin{cases} \alpha s_i/s_i^r, & \text{for } s_i \leq s_i^r \\ \alpha + 2/5(\beta - \alpha)(s_i - s_i^r), & \text{for } s_i^r \leq s_i \leq s_i^a \\ \beta + (5 - \beta)\frac{s_i - s_i^a}{5 - s_i^a} & \text{for } s_i^a \leq s_i\,. \end{cases} \tag{3.1}$$

The parameters α and β of the scaling function are the values of the scaling function at the reservation and aspiration points, respectively. For example, $\alpha = \sigma_i(s_i^r) = 2$ and $\beta = \sigma_i(s_i^a) = 4$. These values will be varied for various labels l_i in order to take into account the preference structure on the importance of labels.

Based on the important classes of report criteria proposed in section 1.5, it is possible to refine the selection operator. The parameters of the scaling function should change so that it returns smaller values for the more important labels. The rationale for such a design is as follows: if a label is least

important, then the agent should be satisfied if the value of the strength of that label is only at its lower reservation level. On the other hand, if another label is most important, then the agent will have a similar level of satisfaction as for the least important one, only if the strength for that label is at its higher aspiration level. As reservation and aspiration levels depend on the distribution of strengths for a label, this reasoning is valid even if different labels have different strength distributions.

Following [180], the following values of the parameters of the scaling function for various importance classes of labels have been proposed in table 3.5.

Table 3.5 Parameter values for scaling function depending on importance class of the label

Importance class	Least imp.	Less imp.	Important	More imp.	Most imp.
α (value of scaling function at reservation level)	3	2.5	2	1.5	1
β (value of scaling function at aspiration level)	5	4.5	4	3.5	3

3.5.3.5 Applying the Proposed Operators Trust Management Systems for Internet Auctions

As described in section 3.5.3.2, the proposed operators should be used to solve two main problems: determining a trust recommendation about a previously unknown agent, and ranking a set of agents (for example, sellers). Let us describe how the operators can be applied for such a purpose.

Assume that a buyer A in an Internet auction considers a previously unknown seller B of a certain product. The trust management system of the Internet auction stores a set of proofs $\{P^{X_i B}\}$ about B. The system can directly use the aggregation operator to produce a single recommendation about the agent, $P^{AB} = \oplus(\{P^{X_i B}\})$.

If the buyer considers more than one seller and wishes to produce a ranking of a set of sellers, $\{B_i\}$, the trust management system can iteratively use the selection operator: $P^{AB_{i_1}} = \triangleleft(\{P^{AB_1}, \ldots, P^{AB_n}\})$, $P^{AB_{i_2}} = \triangleleft(\{P^{AB_1}, \ldots, P^{AB_n}\} \setminus \{P^{AB_{i_1}}\})$ and so on, until a sufficient number of agents (or all) are ranked.

3.5.3.6 A Comparison with eBay's DSR Scale

The Detailed Sellers Rating (DSR) is a new system introduced by eBay for buyers to rate sellers using 4 different criteria:

- product description accuracy,
- communication between seller and buyer,
- shipping time,
- shipping charge.

Buyers can leave one to five stars for each criteria. The DSR system could therefore be expressed using the computational trust representation proposed in this paper for Internet auctions. However, we propose to use a different list of criteria. The DSR criteria are marked on Figure 3.4 with stars near the complaint type. In comparison to our model, the criteria used by DSR cover most of the user-related problems but do not take into consideration item-related complaints (only one criterion concerns the item description). Moreover, DSR does not introduce a context for the report, like the price range or product category.

Although system design is easy to use for buyers, eBay does not instruct them how to use this system. This results in misunderstandings of the proposed rating scale. Also, eBay expects very high average rating results for all criteria for sellers. Therefore, when a buyer submits a report that has a valuation of four stars on some scale, she may think that the report represents a positive feedback; however, to the seller the report is really negative, because the expected averages for higher seller status are above 4.

Moreover, there exist concerns that the introduction of the DSR system together with the new status hierarchy for sellers can have an adverse affect on low-volume sellers.

Another, more significant difference lies in the processing of the reports. eBay does not compute any value beyond the simple averages of received reports for each criterion (and the number of reports. A report that contains valuations for only a subset of the criteria is counted only for these criteria). This means that buyers who want to compare sellers based on the DSR criteria must compare vectors of values together with information on the number of reports for each value. This is a complex task for an ordinary auction user. Using the system proposed here, it is possible to create rankings of sellers (using the selection operator).

The Detailed Sellers Rating system is also unable to take into consideration a preference relation of the buyers on the proposed criteria. All four DSR criteria are considered as equally important. On the other hand, our system takes into account such a preference relation and allows the auction user to specify his own subjective preferences on the criteria.

3.5.4 Computational Trust Models for Credibility Trust

The proposed computational trust representation can be used in other applications other than Internet auctions. A potential application are recommendation systems, where computational trust is used to evaluate recommendations

received from other agents. This kind of application uses trust in the context of the credibility of received opinions. Credibility trust is used in various applications such as epinions, FilmTrust, and many others. Research on credibility trust is abundant [118, 54, 57].

However, credibility trust is rarely treated as a separate context of trust that coexists with other contexts. This is because in many recommendation systems, trust is only used in the context of credibility. On the other hand, in other systems, credibility trust is treated as additional information that is used to process computational trust in the main context [146].

Also, previous research has used simple, scalar representations of credibility trust. The reason for this choice is that credibility trust has been usually acquired automatically through a comparison of received opinions with the trustor's first-hand knowledge. This comparison is often elaborate [54, 146], but results in a scalar real number.

In comparison with computational trust used in Internet auctions, credibility trust is based on completely different information. Trust management systems for Internet auctions are based on reports, while credibility trust is based on observations. The reason for this is that in Internet auctions, agents do not have the opportunity to observe other agents' transactions and must base their evaluations on indirectly acquired proofs. In recommendation systems, agents can compare their own opinions about objects with the opinions received from others. This comparison is the basis of first-hand proofs about the credibility of other agents. However, a few precautions ought to be taken into account when basing credibility trust on observations derived only from the comparison of opinions:

- a similarity of opinions about known objects may not imply a similarity of opinions about newly observed objects,
- a human trustor may consider another agent as credible even if received opinions are not similar,
- a comparison of opinions may be robust to small variations in agents' opinions.

Credibility trust can also be treated as a social opinion that can be elicited by directly asking human users of the system. Credibility trust has all the properties of trust discussed in section 2.2. A simple way of eliciting an opinion about credibility trust is to ask a human agent, A, a question about her credibility trust in B that can be answered on a discrete scale (for example, of five levels).

Elicited credibility trust recommendations can be aggregated with observations obtained from the trust management system by comparison of received opinions with first-hand knowledge, and can be propagated. It is noteworthy that trust in the context of credibility can be propagated transitively. Moreover, when it is necessary to evaluate the credibility trust of a previously unknown agent, the trustor should aggregate available credibility trust recommendations (that have been propagated transitively). In this case, it is

possible that the trustor would receive two extremely different opinions from two equally trusted agents. It is difficult to reconcile these opinions using a scalar computational trust representation.

For these reasons, the more complex computational trust representation using labels may be applied to credibility trust. In this case, labels may represent exclusive levels of credibility, such as: *least credible* (l_1), *less credible* (l_2), *credible* (l_3), *more credible* (l_4), *most credible* (l_5). The associated strengths may be normalized, so that $\sum_{i=1}^{5} s_i = 1$. Notice that given such a definition of labels and strengths, a proof that has equal strengths (equal to 0.2) expresses a neutral value (lack of trust or distrust). A proof with strength values below 0.2 for l_1 or l_2 expresses credibility trust, while a proof with strength values above 0.2 for l_3, l_4 or l_5 expresses trust.

Following Yager [187], Sabeter-Mir [146] defines operators for the aggregation of such proofs. Sabeter-Mir's system also includes a representation of credibility trust as a scalar that is used in the proposed aggregation. The new proofs for credibility trust can be processed differently, using the following credibility trust determination procedure:

1. elicit a human user's opinion about the credibility of a trustee, resulting in a credibility trust recommendation,
2. if available, compare the opinions received from the trustee with the trustor's opinions, resulting in an observation of credibility trust in the trustee,
3. aggregate the recommendation and observation obtained in previous steps, if available, producing a new credibility trust recommendation,
4. if no information is available, use a computational trust propagation algorithm to produce a credibility trust recommendation about the trustee,
5. aggregate the recommendations produced in steps 3 and 4, resulting in a final credibility trust recommendation.

The proposed procedure requires repeated use of the aggregation operator, as well as a comparison of opinions in step 2 and the application of a computational trust propagation algorithm in step 4. Computational trust propagation algorithms will be discussed in the next section; here, it remains to define a new operator (the *composition operator*) of the proposed proofs that will be used by such algorithm.

3.5.4.1 Determining Credibility Trust Observations from Differences of Opinions

Step 2 of the above procedure states that an observation about credibility trust can be created on the basis of a comparison of the opinions received from the trustee with the trustor's own opinions. This comparison must be based on a measure of difference of opinion. Several approaches for trust management in recommendation systems have proposed various such

measures. In order to give a short overview, assume that there are two proofs, $P_1 = \{c, \{l_i, s_i^{P_1}\}_{i=1}^n\}$ and $P_2 = \{c, \{l_i, s_i^{P_2}\}_{i=1}^n\}$. Note that these proofs need not be proofs about credibility trust, but can be made in any context in which two agents, A and B, can encounter each other. In such an encounter, agents can exchange proofs about any third agent, C. After the encounter, each agent may compare the proofs about C received from the other with his own observations that represent what he considers as certain. The sought measure of difference of opinion, $\delta(P_1, P_2)$, is a function of the strengths of the two proofs.

Sabater-Mir [146] proposes to use two values, the total absolute difference of strengths $\sum_{i=1}^{n} |s_i^{P_1} - s_i^{P_2}|$, and the difference of centers of mass, $1/(n-1)\sum_{i=0}^{n-1} i|s_i^{P_1} - s_i^{P_2}|$. The difference measure δ is then a function of the two values.

Golbeck [85] proposes to use average absolute differences calculated on all opinions (strengths) or on the set of extreme opinions. Golbeck proposes to use the top 20% and the bottom 20% of opinions. A final value that is used to determine the difference measure is the maximum difference of opinions.

Another measure of difference of opinion can be based on the Kemeny-Snell measure that is widely used in the comparison of social opinions. The Kemeny-Snell measure can be applied to opinions that are in the form of a preference ranking. Observe that a proof P_1 can be represented as a preference ranking matrix $r_{ij}^{P_1}$ $(1 \leq i, j \leq n)$ as follows.

$$r_{ij}^{P_1} = \begin{cases} 1 & \text{if } s_i^{P_1} > s_j^{P_1} \\ 0 & \text{if } s_i^{P_1} = s_j^{P_1} \\ -1 & \text{if } s_i^{P_1} < s_j^{P_1} . \end{cases}$$

The Kemeny-Snell measure of difference of opinions is given by:

$$\delta(P_1, P_2) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} |r_{ij}^{P_1} - r_{ij}^{P_2}|. \tag{3.2}$$

The proposed measure has several desirable properties, such as the triangle inequality, independence of a similar permutation of opinions [77, 95]. Its values are nonnegative integers, and $\delta(P_1, P_2) = 0$ only for proofs that have identical preference ranking matrices. The value of the Kemeny-Snell measure for two proofs that have n labels is bounded above by $n(n-1)$ (because the maximum of this measure is for two proofs that have completely reversed orderings of labels). A good property of the proposed difference measure is its robustness to small variations in proof strengths. Values of $\delta(P_1, P_2)$ depend only on the ordering of labels according to strength values. This allows us to use the measure regardless of whether proofs have normalized strengths, or not.

It remains to propose how the observation of credibility trust should be created based on the calculated measure of difference of opinion, δ. Firstly, observe that it is necessary to calculate a summary measure of difference of opinion between two agents, A and B. This difference of opinion ought to concern only first-hand knowledge of A and B expressed by observations about other agents. This is because a difference of opinion calculated from recommendations received from third parties would include factors that have no connection to the credibility of B as perceived by A. Let us therefore denote the set of proofs that B has sent to A and that represent B's first-hand knowledge by $O_B = \{P^{BX_i}\}_{i=1}^{m}$ (all P^{BX_1} are observations). Out of this set, we should consider a subset of proofs that concern agents X_i that A has first-hand knowledge about: $O_{BA} = O_B \cap O_A$, where $O_A = \{P^{AX_i}\}_{i=1}^{l}$ and all P^{AX_i} are observations. Let $k = |O_{BA}|$ be the number of observations that A and B have in common with other agents.

l_5 (most credible)	$I_5 = [0, \frac{1}{5}n(n-1)]$
l_4 (more credible)	$I_4 = [\frac{1}{5}n(n-1), \frac{2}{5}n(n-1)]$
l_3 (credible)	$I_3 = [\frac{2}{5}n(n-1), \frac{3}{5}n(n-1)]$
l_2 (less credible)	$I_2 = [\frac{3}{5}n(n-1), \frac{4}{5}n(n-1)]$
l_1 (least credible)	$I_1 = [\frac{4}{5}n(n-1), n(n-1)]$

Fig. 3.7 Intervals of the Kemeny-Snell difference measure for determination of credibility trust level

To create a credibility trust observation P_{cred}^{AB} on the basis of a comparison of first-hand knowledge of A and B, a histogram of the observed differences between the opinions of A and B can be created. Let $d_i = \delta(P^{AX_i}, P^{BX_i})$. Since $0 \leq d_i \leq n(n-1)$, where n is the number of labels in P^{AX_i}, it is possible to create five intervals that correspond to the five labels of a credibility proof. These intervals could be created by equally dividing the interval $[0, n(n-1)]$ into five proportionate parts, as shown in Figure 3.7.

Each value $d_i \in I_j$ will increase the count of values m_j for the interval j by 1. After all observed differences are counted, the values m_j will be normalized (divided by the number of observations, k). The normalized values $m_j^{'} = m_j/k$ form an empirical histogram of the differences of opinion between A and B. However, in order to determine the strengths of the credibility observation P_{cred}^{AB}, it is good to compare the values $m_j^{'}$ to a reference level. This would allow us to take into account how human agents interpret and use the computational trust representation. For the reference levels, the values of a histogram created for all observed differences of opinion (not just for the agent B) can be used. Agent A can maintain a second histogram, $\overline{m}_j$, that is created as described above, but from a comparison of the observations of A with proofs received from all other agents. The comparison of the two histograms can determine the strengths of P_{cred}^{AB}.

A piecewise linear scaling function may be used to compare the histogram values $m_j^{'}$ and $\overline{m}_j$. This function can be given by:

$$\sigma(m_j^{'}, \overline{m}_j) = \begin{cases} \frac{1}{5} m_j^{'} & \text{if } m_j^{'} < \overline{m}_j \\ \frac{1}{5(1-\overline{m}_j)}(4m_j^{'} + 1 - 5\overline{m}_j) & \text{if } m_j^{'} \geq \overline{m}_j. \end{cases}$$

The proposed scaling function has values $\sigma(0, \overline{m}_j) = 0$, $\sigma(\overline{m}_j, \overline{m}_j) = 0.2$ and $\sigma(1, \overline{m}_j) = 1$ for any $\overline{m}_j$. The values of the scaling function for all j add to 1 ($\sum_{j=1}^{5} \sigma(m_j^{'}, \overline{m}_j) = 1$), therefore the values of $\sigma(m_j^{'}, \overline{m}_j)$ can be used directly as the strengths of the proof P_{cred}^{AB}. If the differences between the opinions of A and the opinions of all other agents have the same distribution as the differences of opinions between A and B ($\overline{m}_j = m_j^{'}$ for all j), then the credibility trust observation P_{cred}^{AB} will have strengths equal to 0.2. This property ensures that the credibility trust observation will be neutral in such a case.

3.5.4.2 Aggregation Operator for Credibility Trust Proofs

Following Yager [187] we define the aggregation operator for credibility trust proofs as follows.

$$P_{cred_agg}^{AB} = \oplus_{cred}(P_1, \ldots, P_n),$$

where $P_i = \{\text{credibility}, \{l_j, s_{i_j}\}_{j=1}^{5}\}$ are proofs about the credibility trust of B. Note that these proofs could have been obtained in any of the steps of the credibility trust determination procedure 5. The proofs could therefore be observations of A or recommendations about the credibility trust of B received from any other agent. The received recommendations would have to be composed with proofs about the credibility of their source; this composition will be described in the next section.

The result of the aggregation operator, $P_{cred_agg}^{AB} = \{\text{credibility}, \{l_j, s_j\}_{j=1}^{5}\}$, can be defined using formulas for the strengths s_j. Yager has proposed two methods for aggregating strengths of belief. Both of these methods have desirable properties such as symmetry, monotonicity, associativity and the preservation of the neutral proof as a unity. The first, simpler method is a weighted product of the strengths of contributing proofs:

$$s_j = \frac{\prod_{i=1}^{n} s_{i_j}}{\sum_{j=1}^{5} \prod_{i=1}^{n} s_{i_j}}. \tag{3.3}$$

This method is undefined for conflicting proofs such that $s_{1_j} s_{2_j} = 0$ for all j. In this special case, the strengths of the aggregated proofs are all equal and equal to 0.2 (the aggregated proof is a neutral proof).

The second method defined by Yager is based on the following procedure. First, for each label l_j of a credibility trust proof, the set $\{s_{i_j}\}$ of contributing strengths is partitioned into three subsets: the strengths equal to 0.2

(neutral), the strengths above 0.2 and the strengths below 0.2. The last two subsets can be defined as: $L = \{s_{i_j} : s_{i_j} < 0.2\}$ and $H = \{s_{i_j} : s_{i_j} > 0.2\}$. Let us denote the sizes of the two subsets by $n_L = |L|$ and $n_H = |H|$. The formula for aggregated strengths is given by:

$$s_j = Q(\frac{n_L}{n_H})T(L) + (1 - Q(\frac{n_L}{n_H}))S(H). \tag{3.4}$$

Here, Q is a nondecreasing function on the interval $[0, 1]$. For simplicity, we shall assume that Q is the identity function. T and S can be chosen from a large set of operators, but a simple and intuitive choice is $T(L) = \min(L)$ and $S(H) = \max(H)$. The formula for aggregated weights simplifies then to the following expression:

$$s_j = \frac{n_L}{n_H} \min(L) + (1 - \frac{n_L}{n_H}) \max(H).$$

3.5.4.3 Composition Operator for Credibility Trust Recommendations

The difference of the approach for representation and processing of credibility trust and of the approaches proposed by Yager and Sabater-Mir lies in the treatment of transitive composition of credibility trust. The composition operator formalizes the observation that credibility trust is inherently transitive: if an agent A has a high credibility trust in an agent B and receives from B a recommendation that expresses high credibility trust in a third agent C, then A will most likely believe C.

Let us define the composition operator as follows: $P_{compose} = \odot(P_1, \ldots, P_k)$. A special case is $k = 2$: $P_{compose} = \odot(P_1, P_2)$. The following intuitive requirements can be specified for a composition operator of computational trust: if P_1 expresses high trust, then $P_{compose}$ should be similar to P_2. On the other hand, if P_1 expresses high distrust, then $P_{compose}$ should express neutral trust.

Given the above definition of labels and strengths, it is possible to define a function that calculates the overall credibility trust rating of a proof. This can be done as follows:

$$e^T(P) = 1/3 \sum_{i=3}^{5} (i - 2)s_i,$$

for $P = \{c, \{l_i, s_i\}_{i=1}^5\}$. If $s_5 = 1$ for a certain proof, P_{TT}, (and consequently, strengths for the other labels are zero), then $e^T(P_{TT}) = 1$. For a proof P_{HT} such that $s_4 = 1$, $e^T(P_T) = 2/3$. For a proof P_T such that $s_3 = 1$, $e^T(P_T) = 1/3$. For all proofs such that $s_1 + s_2 = 1$, the value of $e^T(P)$ is zero.

The function $e^T(P)$ can be used to define the composition operator in the following manner. Let $P_{compose} = \{c, \{l_i, s_i^{P_{compose}}\}\} = \odot(P_1, P_2)$, where $P_1 = \{c, \{l_i, s_i^{P_1}\}\}$ and $P_2 = \{c, \{l_i, s_i^{P_1}\}\}$. Then:

$$s_i^{P_{compose}} = (1 - e^T(P_1))1/5 + e^T(P_1)s_i^{P_2}. \tag{3.5}$$

This transformation of the weights of the two proofs P_1 and P_2 preserves their normalization. To see why this is so, consider that $\sum_{i=1}^{5} s_i^{P_{compose}} = 1 - e^T(P_1) + e^T(P_1) \sum_{i=1}^{5} s_i^{P_2} = 1$.

Then, the general composition operator can be defined by repeatedly applying the composition operator for two proofs: $\odot(P_1, \ldots, P_k) = \odot(\odot(P_1, P_2), P_3, \ldots, P_k)$.

The proposed formula for the composition operator has the undesirable property that when a neutral proof (or even total distrust) will be combined with a sequence of proofs of total trust of increasing length, the resulting proof will approach a limit of total trust. This property is undesirable because a composition of a sequence of proofs of increasing length should not converge to total trust (especially if the first proof was total distrust), but rather to the neutral proof. In order to achieve such a property of the composition operator, it is possible to modify equation 3.5 as follows:

$$s_i^{P_{compose}} = (1 - (e^T(P_1))^k)1/5 + (e^T(P_1))^k s_i^{P_2}, \tag{3.6}$$

where k is a parameter. An exponent of $k = 4$ results in a fast convergence of a sequence (composed of an initial neutral proof and subsequent proofs of total trust) to the neutral proof.

3.6 Algorithms and Protocols of Computational Trust and Reputation Management

3.6.1 Establishing Initial Trust

A trust management system operates on the basis of proofs and is capable of recommending computational trust valuations based on proofs available about the trustee and other agents. However, sometimes the TM system does not have any proofs available about the agent. This situation is referred to as establishing initial trust (and should not be confused with the situation where there are no proofs that represent direct observations of the behavior of agent B made by agent A. Initial trust needs to be established if the TM system has no information at all about B).

At first, the problem seems insurmountable. However, it is possible for the TM system to calculate a general trust towards all agents. This general trust value is sometimes called the *trust propensity*, and represents the TM system's view of the environment it operates in [143].

This approach can be refined if all agents can be partitioned into groups. If the TM system would have any proofs about the past behavior of agents in a group, it could compute a group reputation measure. However, let us

consider a situation where no reports are available about the agents in a group. In such a case, it is possible to use stereotypes to determine initial trust [97]. In order to apply stereotypes, the groups of agents should be defined using some observable attributes of agents (for example, all agents that live in Tokyo and work for the government). A stereotype is then a prior assumption about the trust in the agents belonging to the group. Stereotypes can be represented as recommendations that are not made for a specific agent, but rather for a hypothetical agent that matches the stereotype. Applying stereotypes is really a form of similarity propagation where the similarity relation is extended to include stereotypes (see section 2.2.7).

3.6.2 Gathering and Processing of Proofs

The gathering and processing of proofs is one of the most important parts of the TM system. In the system proposed in this chapter, this task is delegated to the Proof Discovery Protocol (PDP. See section 3.2).

A basic distinction of TM systems into centralized and distributed systems also assumes that in the first case, all proofs will be available from a single agent, while in the second, every agent has his own set of proofs, but agents can share the proofs among each other. Both of these cases can be implemented using the PDP. The distributed system can also support a hybrid architecture, where large sets of proofs are stored by a few special agents (as in a P2P network with superpeers).

TM algorithms can be accordingly partitioned into algorithms that rely on global knowledge or algorithms that rely on local knowledge. On the other hand, some global knowledge algorithms can also operate on a subset of proofs available to an agent at the time of algorithm execution. The real distinction between algorithms with respect to availability of information is whether or not the algorithm can control the process of discovering necessary proofs during its execution. If it cannot, we can speak of an algorithm that relies on the information available at execution time (which can be local or global knowledge). If it can, then the algorithm is more suitable for a distributed system because it can start with local knowledge and then attempt to extend it using the PDP.

For the first type of algorithms, the PDP can work continuously and gather as many proofs as possible so that when the algorithms are executed, it will have sufficient proofs available. For the second type of algorithm, the PDP may be invoked by the algorithm itself, whenever it needs additional proofs. However, the most extreme form of distribution would require that the PDP cannot work independently of the TM algorithm and is invoked during every iteration of the algorithm to find a proof. This approach is likely to incur a large communication overhead and to delay the algorithm's operation. In the TM system proposed in this chapter, the PDP works continuously, but also supports queries from algorithms that can result in the search for required proofs (for example, for more proofs about a previously unknown agent).

Examples of completely distributed algorithms that search for proofs in every iteration are TM algorithms used in P2P networks, such as Eigentrust, that will be described below.

3.6.3 Trust Propagation Algorithms

Human trust propagation is one of the principal properties of trust that can be exploited by TM methods. As described in section 2.2.7, there can be many types of trust propagation, but they all have the same effect: the establishment of new trust (or distrust) relations to strangers. Human trust propagation usually relies on additional information obtained by human agents. In a TM system, such information (represented as proofs, or computational trust) can be automatically processed. This processing is done by Trust Propagation Algorithms. A result of a Trust Propagation Algorithm can be a recommendation for a human user that leads to human trust propagation. On the other hand, if agents are not humans, then the results of Trust Propagation Algorithms can be processed automatically.

This section gives a classification of Trust Propagation Algorithms according to the specific problem that they solve. Next, the section presents an overview of such algorithms presented in the literature. A new Trust Propagation algorithm is proposed and evaluated. The section concludes with a discussion of general trust propagation operators that can be used to decompose Trust Propagation Algorithms.

3.6.3.1 Computational Trust Propagation Problems

The trust propagation algorithms described in the literature solve at least three different types of computational trust propagation problems. These problems will be defined and discussed in this section for the sake of better explanation of the algorithms.

We will define four kinds of computational trust propagation problems:

- Problem 1. Given TR (DTR) and two agents, Alice (A) and Bob (B), such that no edge $e_{AB} = (A, B)$ exists in T (or DT), establish a new edge e_{AB} and its strength s_{AB}.
- Problem 2. Given TR (DTR) and an agent A, find a set of k agents in V that are most trusted (distrusted) by A. A variant of Problem 2 requires that the produced set of agents should be sorted by trust (distrust) of A or requires that the entire set of agents V should be ranked by trust or distrust of A.
- Problem 3. Given TR (DTR), create as many new trust (distrust) edges as possible between pairs of agents in V. The output of an algorithm that solves Problem 3 is a new trust (distrust) relation TR^P (DTR^P) that includes the new propagated trust edges. The new trust edges may be established on the basis of a recursive or iterative procedure that uses the

trust edges from T (DT) as well as new trust edges created in a previous iteration.
- Problem 4. Given TR (DTR) and an agent A, determine a measure of trustworthiness of A based on the propagation of computational trust of a set of other agents towards A. The considered set of agents can be all agents in V.

Note that a solution for Problem 1 can be used to solve problem 3, if it is applied iteratively to several pairs of agents that are not connected by edges in TR (DTR). However, such an approach may not be computationally efficient. On the other hand, a solution for Problem 3 may also contain a solution for Problem 1. Also, a solution to Problem 3 may be used to produce a ranking of trusted agents for any agent A, and therefore can be used to construct a solution to Problem 2.

Problem 4 is a special kind of propagation problem that is related to the notion of *global trust or reputation*. Recall from section 2.1 that trustworthiness is a property of an agent that is not subjective with regard to the trustor (in other words, it is not a relational property, such as trust). Global trust or global reputation are terms used in the literature for measures of trustworthiness. Here we can show that computing these values is not different in principle from other kinds of computational trust propagation. A following algorithm can be used to solve Problem 4:

1. Choose a propagation scope r (the maximum length of a propagation path in T).
2. Starting at A, do a BFS search to find the set X of all nodes within distance r from A.
3. For all nodes $B \in X$, solve Problem 1 for the pair of nodes B and A.
4. Aggregate the propagated computational trust strengths for all nodes $B \in X$.

The last step of the proposed solution for Problem 4 is the only one that seems to require a specialized approach. However, we will show that this step is in fact a basic step of computational trust propagation algorithms for Problem 1 or Problem 3. Thus, Problem 4 is not different in principle from the other kinds of computational trust propagation problems. The single drawback of the proposed approach may be the inefficiency of repeatedly solving Problem 1; this may be avoided by specialized algorithms that integrate steps 2 and 3 of the proposed procedure.

3.6.3.2 Correctness of Trust Propagation Algorithms

The problem formulations described above have one serious drawback. Any solution for these problems could be proposed that does not necessarily fulfil requirements of correctness for trust propagation.

The obvious approach to evaluate the correctness of computational trust propagation algorithms would be to test them on human subjects and evaluate whether the subjects think that the recommended, propagated trust relations are valid (in case of algorithms for Problem 1 and Problem 3), or whether the ranking produced matches the trust preferences of humans subjects (Problem 2). Unfortunately, such an approach has not yet been described in the literature. The main reason may be the difficulty of performing such experiments; the design of such experiments requires knowledge and training in experimental games. Another evaluation approach that has been used in previous work is based on the availability of declarative trust networks that contain trust relations declared by human subjects. In such a trust network, a certain number of trust edges can be "hidden" and constitutes a test set. The computational trust propagation algorithm can use the remaining trust edges. The output of the algorithm is then compared against the trust edges in the test set.

The correctness of an algorithm for Problem 1 can be simply defined using this evaluation approach. An edge between two agents A and B is "hidden" and the algorithm is correct if it can reconstruct an edge of similar strength. An algorithm for Problem 2 can be tested as follows: k strongest trust edges originating in a certain agent A are "hidden". The algorithm is used to produce a ranked set of k agents that A can trust. The order of these agents is compared with the order of the trust edges in the test set. The correctness of an algorithm for Problem 3 can be tested by "hiding" a random subset of trust edges. The algorithm will return a result set of propagated trust edges. The result set can be compared with the test set. It is possible to evaluate the recall (percentage of the edges in the test set that are also in the result set) of the algorithm. However, the algorithm's precision cannot be evaluated because of the fact that not all real trust relations may have been declared by the human subjects in the experimental trust network.

Also, it is necessary to keep in mind that all tests using trust networks are dependent on the structure of the chosen network. Preferably, an algorithm should be tested on several trust networks of different origin.

A propagation algorithm for Problem 2 or Problem 3 can compute new strength values for existing edges in the trust network. The difference in strengths between existing trust edges is another measure of the algorithm's correctness. We refer to this measure as the *consistency* of the algorithm. Intuitively, a high consistency indicates that the trust propagation algorithm does not produce recommendations that are in conflict with the first-hand knowledge of an agent.

A final, important correctness measure of an algorithm for Problem 2 or 3 is the amount of new trust edges that the algorithm recommends. We refer to this measure as the *edge growth* of the algorithm. Edge growth forms a tradeoff with the recall of a propagation algorithm, because if an algorithm will add many edges to a starting node, the likelihood that it will rediscover a removed edge grows. In a way, edge growth is related to the precision of

the algorithm; however, the exact precision is impossible to evaluate without additional data about which of the new trust edges are correct.

Our correctness measure can be formally specified as follows: we are given a trust network TR. In this network, we shall select nodes A that will be used to find user-centric subnetworks of TR. These subnetworks will be the test cases of the algorithm evaluation method. Let the set of all test cases be $H = \{(A_i, G_i =< V_i, E_i >, re_i), i \in 1, \ldots, n\}$, where A_i is a starting node in TR, G_i is the subgraph of TR within radius r of A_i (including all edges between the nodes within the radius), and re_i is the "hidden" edge removed from G_i. Let the output of a propagation algorithm on all test cases be $O = \{G_i^P =< V_i, E_i^P >\}$, where G_i^P is a propagated subgraph consisting of the same vertices as G_i, but with new (propagated) edges E_i^P with strengths s_e^P.

Assume that the propagated trust edges in E_i^P are ordered according to their decreasing strengths. Let us denote the set of the strongest l edges in E_i^P as lE_i^P. This set may or may not include the removed edge re_i. Our correctness measure is referred to as the *total inclusion measure* (TIM):

$$TIM(l) = |\{i : re_i \in lE_i^P\}|/n.$$

The total inclusion measure is equal to the percentage of test cases in which the l strongest propagated edges included the "hidden" edge. If TIM is equal to 100%, then the set of the strongest l propagated edges includes the removed edge re_i.

The set lE_i^P may include other edges besides the removed edge and the original trust edges. The value of $\lambda = \frac{l}{|E_e|}$ indicates how many additional edges have been included in the set of propagated edges. While some of these edges may point to new, trustworthy agents that have been strangers until now, a high value of λ increases the likelihood that untrustworthy agents are recommended by the trust propagation algorithm. λ is therefore similar to edge growth. It is possible to define TIM as a function of λ: $TIM_\lambda = TIM(\lambda|E_e|)$. A good trust propagation algorithm would therefore have a high total inclusion measure TIM_λ for a low λ. We shall investigate the relation between these two values for GKRT and for CloseLook.

3.6.3.3 Computational Trust Propagation Algorithms

Advogato

The oldest known computational trust propagation algorithm is Advogato [91, 90], an algorithm for solving Problem 2 for a trust network TR (Advogato is not suitable for distrust propagation). However, it does not produce a ranked set, but returns an unordered set of k fully trusted agents. Advogato was developed for the purpose of public key distribution. Given the existence of a set of authorities that are fully trusted a priori, Advogato can be used to

determine all nodes that can be trusted because they are sufficiently trusted by the authorities. Advogato can calculate the set of k trusted agents not just for one source A, but for any set of authoritative agents.

Advogato uses the Floyd and Fulkerson algorithm for maximum integer network flow computation. In order to do this, Advogato needs to transform the trust network TR into a flow network. This transformation is the essence of the algorithm. Advogato assigns capacities to nodes in V based on their distance from the source A (or the shortest distance from multiple sources) and the outdegree of nodes with a smaller distance (if any). The initial capacity of the source A is k (for multiple sources, each receives the same capacity). Next, the algorithm calculates an average outdegree d_j of all nodes at distance j from the sources. The capacity of all nodes at distance $j + 1$ is the total capacity of their neighbors at distance j divided by d_j.

After the capacity calculation, Advogato transform TR into a flow network by changing each node $v \in V$ into a pair of nodes, v^+ and v^-. An edge is added from v^- to v^+ with a capacity equal to the capacity of v. The maximum integer network flow is found in the transformed network, and all nodes that are reached by a non-zero flow from the sources are returned as trusted nodes.

The main goal of Advogato's design has been attack resistance. The authors formulate a so-called "bottleneck property" of computational trust propagation algorithms which states that the strength of a propagated trust edge from A to B should not be affected by the changes in the trust network among the successors of B. If this property holds, B cannot manipulate his own trust score by issuing his own recommendations. Advogato has the "bottleneck property" and is therefore deemed attack-resistant against adversaries that can create trust edges among themselves.

The main drawback of Advogato is that it only produces a set of fully trusted agents, without a ranking that would allow us to further discriminate among these agents. Interestingly, Advogato uses transitive trust propagation, but the composition of trust is based on the minimum of trust strengths along a propagated path, not on the multiplication of these strengths. The reason for this is the application of the maximum integer network flow algorithm.

Appleseed

Appleseed is another algorithm for Problem 2 [196] (again, only for trust and not distrust propagation). Appleseed avoids the drawbacks of Advogato by producing a ranked set of most trusted agents. Similarly to Advogato, Appleseed applies an established algorithm for computational trust propagation. In this case, it is the spreading activation model by Quillian. In this approach, energy is propagated among nodes of a graph. Edge strengths represent capacity of energy transmission. The higher the strength, the more energy will flow to the next node. Yet, energy can only flow from a node if this node

receives a sum of energy that exceeds a threshold (the node is then activated – hence the name). Energy is divided among successor nodes based on normalized local edge strength. This is a further advantage of this approach over the maximum integer network flow calculation which requires global knowledge. The Appleseed algorithm, unlike Advogato, is suitable for distributed implementation.

In contrast to Advogato, Appleseed uses transitive trust propagation that is multiplicative. The algorithm modifies spreading activation in several ways. Firstly, energy transmission in the spreading activation model can be for an unlimited distance. Appleseed introduces *trust decay* for transitive propagation. In this approach, not all energy (trust) is propagated to successor nodes, but only a fraction of it. This fraction is controlled by the spreading factor d. Appleseed uses a spreading factor of $d = 0.85$, meaning that 15% of energy is retained in the node, and the rest is propagated. Another modification is introduced to handle dead-ends. In spreading activation, if an amount of energy reached a node with zero outdegree, this energy was lost and not propagated further. Appleseed deals with this problem by modifying the trust network. For every node, it introduces a special edge of full strength (equal to 1) that leads back from this node to the source A. As a result, energy flows back if a dead-end is reached and can be redistributed.

The edges leading back to the source allow Appleseed to deal with problems related to trust normalization that will be discussed next.

Normalization in Trust Propagation

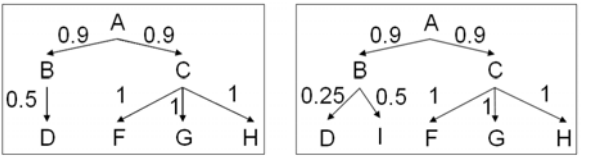
Appleseed is the first discussed trust propagation algorithm that introduces normalization of edge strengths. Generally, normalization is often introduced because of several reasons. For some trust networks, edges can be made arbitrarily strong. On the other hand, if there is a limitation of trust strength (for example, only allowing trust strengths to be in the range of $[0, 1]$) then this limitation may be easily exceeded by trust propagation. Consider multiplicative, transitive computational trust propagation as used in Appleseed. Imagine that A trusts B and C with strength 1, while B and C both trust D with strength 1. Then it is easy to see that A should trust D with strength 2, unless trust strengths are normalized. An even more serious trust normalization problem occurs for instances of Problem 3 if the trust network contains cycles. In the same example as before, let us add a trust edge from D to A with strength 1. Then, it is easy to see that we can obtain the following propagated trust edges: from A to D with strength 2, from D to B with strength 2, and as a result from A to B with strength 4. If this process is applied recursively, the strength of propagated trust edges can become arbitrarily large.

There can be various approaches to trust normalization, and these approaches can be adapted to the chosen trust propagation algorithm. Normalization is usually applied on the strengths of trust edges before a trust propagation algorithm is executed, or at each stage of the trust propagation

algorithm before determining the strengths of propagated links. The simplest form of normalization would be to use a linear weighted sum of local outgoing edge strengths:

$$w_{AB} = \frac{s_{AB}}{\sum_{(A,X)\in T} s_{AX}}.$$

However, normalization creates its own problems. Consider the example 1 of a trust network shown in figure 3.8:



Example 1 **Example 2**

Fig. 3.8 Examples of trust networks with normalization problems

In the network shown in example 1, it is possible to apply multiplicative, transitive trust propagation to determine the trust of A in other nodes (D, F, G, H). If no normalization is applied, then A should trust D with strength 0.45, and the other nodes with strength 0.9. Yet if linear trust normalization is applied, A will trust D with strength 0.5, and the other nodes with strength less than 0.17! This is clearly a paradox, because it would appear that A should trust the nodes F, G and H much more than D, but as a result of normalization it will trust them for less. The reason for this result is that the node C is "punished" by the normalization formula for being "too trusting": C has three equally trusted neighbors, while B (which is equally trusted by A) has only one trusted neighbor. It seems that the linear normalization formula is too strict in discriminating between nodes with various amounts of trusted neighbors. A different approach can use non-linear, for example quadratic normalization:

$$w_{AB} = \frac{s_{AB}^2}{\sum_{(A,X)\in T} s_{AX}^2}.$$

Quadratic normalization (or using any other power) will not change the results of Example 1. Yet, consider the trust network shown in Example 2. In this network, using linear normalization would result in A trusting D with a strength of about 0.17, that is, equally as A would trust F, G and H. With square normalization, A would trust D with strength 0.1, which is less than the trust of A in F, G and H.

Yet a different approach would use Ordered Weighted Averaging, assigning larger weights to edges with a larger strength. In this approach, only the

ordering of the strengths is important; therefore, the edge with the largest strength will get the largest weight, regardless of whether it has a strength of 1 or 0.5. The advantage of this approach is that a user who is reluctant to use high strengths will be treated similarly as a user who uses higher strengths.

All of the previously described normalization approaches are local in the sense that they only consider the outgoing links of a single node. Another approach could use global information; for example, it would be possible to run the trust propagation algorithm without normalization, and then normalize the resulting strengths using a global distribution of trust edge strength. This approach has a certain drawback; if adversaries were to succeed in affecting the trust propagation algorithm in such a way that they would receive higher trust strengths relative to the rest of the network (for example, by using cycles), then the global normalization would still assign them higher trust strengths than the other nodes.

TidalTrust

TidalTrust [54] is an algorithm that takes a more straightforward approach to trust propagation. It is an algorithm for Problem 1, aiming to establish a trust relation between two nodes, A (the trustor) and B (the trustee). TidalTrust considers only multiplicative, transitive trust propagation and is not capable of propagating distrust. The main idea of the algorithm is to select only the most important information from the trust network, disregarding the rest as a source of additional noise and complexity. TidalTrust only considers the best paths in TR leading from A to B. It uses a modified BFS algorithm to find the shortest paths from A to B (considering each trust edge as having length 1). Afterwards, TidalTrust only considers the paths of the shortest length, disregarding others. This approach is motivated by empirical observations from trust networks obtained from recommendation systems which show that the longer the path, the smaller the likelihood of obtaining good recommendations [54].

Among the shortest paths from A to B, TidalTrust calculates the maximum of the minimum edge strengths on a path, s_{maxmin}. The minimum of the edge strengths on a path is therefore considered as a measure of strength of this path, as in Advogato. Yet, after the path selection process, TidalTrust again uses multiplication to calculate the strength of a path from its edge strengths, as in Appleseed. s_{maxmin} is used as a threshold value to select the best paths. The paths that will be considered in the final trust computation are only the shortest paths from A to B with a minimum of the edge strengths that is not less than s_{maxmin}.

The same threshold is also used to change the normalization approach. When the edges are normalized, TidalTrust considers only the outgoing links with a strength that is not less than s_{maxmin}, leading to the following normalization formula:

$$w_{AB} = \frac{s_{AB}}{\sum_{(A,X)\in T: s_{AX} \geq s_{maxmin}} s_{AX}}.$$

Thus, TidalTrust disregards weak edges in the normalization process, focusing only on the strongest ones. After the best paths are selected, TidalTrust normalizes the edge strengths and applies multiplicative transitive trust propagation to all of the best paths.

Matrix Algorithms

It is not difficult to notice that transitive, multiplicative computational trust propagation can be expressed in matrix form. If the trust network T is expressed as an adjacency matrix, then the kth power of that matrix T^k represents the result of propagating all trust relations transitively k times. This observation has been used independently in at least two important computational trust propagation algorihtms: GKRT [57] and Eigentrust [69]. These algorithms are discussed jointly in this section.

The algorithm of Guha et al. (GKRT) is an algorithm for Problem 3. It uses four kinds of trust propagation that can all be expressed in the form of matrix multiplication and transposition. One of these is multiplicative transitive propagation. Another important kind of propagation used by Guha et al. is a kind of similarity propagation called *co-citation*. Co-citation defines the similarity of two agents, A and B, based on these agents position in the trust network itself. If A and B trust similar sets of other agents (the sets of agents connected by outgoing edges from A and B in T have a large intersection), then A and B are considered similar. Similarity propagation is then applied using this similarity relation. Notice that while multiplicative transitive trust propagation for paths of length 2 can be expressed in the form of matrix multiplication T^2, a single trust propagation using co-citation can be expressed using the formula $TT^{\mathrm{T}}T$. The other two forms of propagation used by Guha et al. are *transpose* propagation (referred to as reflexive propagation in this book. See section 2.2.7), and *trust coupling* which occurs if A's trust in B is propagated to C because B and C trust many agents in common. A single reflexive propagation can be expressed by the formula TT^{T}, whereas trust coupling by T^2T^{T}. All of these forms of single propagation can be expressed by the formula TM, where M is a matrix operator. In order to extend the propagation further, it is necessary to use the next powers of the operator M.

The work of Guha et al. is unique in one particular aspect: it is the only work that directly considers the propagation of computational distrust, treating distrust distinctly from trust and taking into account its basic propagation properties 2.2.7. Guha et al. consider a one-step propagation of trust with distrust. On the other hand, Guha et al. also consider the possibility of jointly propagating computational trust and distrust, which requires that the two quantities should be combined (Guha et al. substract computational distrust from trust). Note that while this is a feasible operation for computational trust and distrust, there are theoretical objections to this kind of approach,

because it assumes that humans would compensate distrust with trust (treating both attitudes equally). Interestingly, Guha et al. have found that only a one-step propagation of trust with distrust is validated by empirical data.

Guha et al. were one of the first to propose a method of experimental validation of computational trust propagation through the use of declarative data about a trust and distrust network, as described in this section. Guha et al. used a dataset from the Epinions recommendation system [109] (a Website where users may recommend certain items, such as books or movies, and declare their trust or distrust in the recommendations of other users). Among the considered kinds of propagation, transitive, co-citation and one-step propagation of trust with distrust were the most successful.

The limitations of the Epinions data for the evaluation of computational trust propagation lies in the fact that users can only express binary statements about trust or distrust. The results of a computational trust propagation algorithm have to be rounded in order to compare them with the declared values. Guha et al. employ several rounding techniques, but their analysis makes it impossible to evaluate whether the observed differences are due to rounding or the employed trust propagation algorithms. Guha et al. state that the rounding methods turned out to be critical in getting good results; without using another dataset with trust declared on a continuous scale, it is hard to validate the results of empirical analysis on the Epinions data.

GKRT has one significant drawback: it is vulnerable to cycles, when transitive propagation is employed. Guha et al. attempt to limit this vulnerability by limiting the length of transitive propagation to three steps. However, this still does not entirely avoid cycles, and makes the approach vulnerable to adversaries who can create a large number of short cycles (this is especially true if adversaries can create an arbitrary number of clones with fake identities). This vulnerability is common to all matrix algorithms.

Eigentrust is a trust propagation algorithm that can be easily explained using the previously discussed methods. The basic Eigentrust algorithm uses multiplicative transitive trust propagation and linear normalization. However, Eigentrust does not attempt to solve any of the trust propagation problems described above. Rather, it is an algorithm for calculating a measure of trustworthiness of each agent (which does not depend on the trustor, only on the trustee) 2.1. A measure of trustworthiness is simple to calculate if only multiplicative transitive propagation is used. A result of applying the propagated trust matrix T^n on a vector of uniform a priori trustworthiness is the sought trustworthiness of all agents. This calculation can be simplified by taking the left principal eigenvector of T. Eigentrust modifies the basic algorithm by introducing nodes that are trusted a priori. Trust to these nodes is not propagated, and the calculated trustworthiness is modified so that in each iteration, it takes into account a measure of trust in the pretrusted nodes. This modification has two purposes: first, it attempts to protect the computation against adversaries who can form cycles, and second, it ensures that

the calculation of high powers of the trust matrix T will converge, because the modified matrix will be irreducible and aperiodic.

Furthermore, the Eigentrust algorithm can be distributed and further modified to improve its security in a distributed Peer-to-Peer environment. This modification uses a Distributed Hash Table (CAN) that is also used for gathering proofs.

Sunny

Sunny is the most complex computational trust propagation algorithm among those described in this section. [85]. It is an algorithm for Problem 1, with additional input in the form of recommendation profiles. Sunny is therefore suitable only for computational trust propagation in recommendation systems. Sunny propagates trust in a manner similar to TidalTrust. The difference lies in the selection process of the paths used to calculate trust from A to B. Firstly, Sunny creates a subset of the shortest paths from A to B that are disjointed and do not contain cycles. From these, Sunny selects only a subset of paths that contain specially selected neighbors of B. The selection process of these neighbors relies on the calculation of a confidence interval in the recommendations of B. This confidence interval is calculated using a Bayesian network constructed from the trust network TR. The transformation is made in the following manner: for each node $X \in V$, the Bayesian network contains a variable that represents the belief of X in a recommendation from B. This variable depends on another variable Y if there is an edge between X and Y in T. The Bayesian probability distribution is constructed using a similarity measure of the recommendation profiles of X and Y. After the Bayesian network is constructed, it is used for probabilistic sampling that allows us to determine the upper and lower limits of the confidence of each node in a recommendation from B, using a procedure described in [86].

The calculated bounds on confidence levels depend on additional states of the neighbors of B that can be used for the final calculation of trust. Each neighbor of B has one of three states: "unknown", "include" or "exclude" from the set of paths used for the final trust calculation. In the beginning, the Bayesian network is used to calculate bounds on the confidence of A in B's recommendation when all neighbors of B have an "unknown" state. After this initial step, Sunny iterates through all neighbors of B and tries to change their state to "include" and again calculates confidence bounds for the belief of A in a recommendation from B. If the resulting confidence bounds are too different from the original confidence bounds, then the state of this neighbor of B is changed to "exclude". Finally, only the neighbors of B that do not change the original confidence bounds too much are selected. Only the paths from A to B that end in one of these neighbors are selected for the final trust calculation.

3.6.4 CloseLook: A Limited-Scope Trust Propagation Algorithm

Of the algorithms presented above, few have directly considered computational efficiency and communication resources requirements. An exception is Appleseed that has been designed for distributed implementation [196]. Appleseed achieves this goal because the distribution of energy among the nodes during the algorithms' operation can be simply distributed. However, Appleseed does not consider the communication overheads of such an approach; the amount of asynchronous communication among all nodes, required until the algorithm converges, may be large. Eigentrust is another algorithm that can be distributed, although it relies on a structured Peer-to-Peer network for its scalability properties. However, both Appleseed and Eigentrust have distributed both the task of discovering proofs and the trust calculation (and using the same approach). A distributed computational trust propagation algorithm may use different mechanisms for the tasks of proof discovery and distributed computation.

For example, consider GKRT. This algorithm assumes global knowledge. Yet, in practice, the transitive trust propagation of Guha et al. is limited to a scope of three trust edges (for larger scopes, the computational cost becomes prohibitive). This suggests that GKRT could be distributed and calculated locally by each node, if this node could acquire the knowledge about his neighborhood in the trust network within a radius of three. This task is actually quite simple to accomplish using gossiping protocols that randomly walk the network for a specified distance. A walking message contains information about the known neighborhood of a node and the distances to each neighbor. Whenever a walking message passes another node, this node acquires more knowledge about his neighborhood (and can limit his knowledge to a radius of three). Such a gossiping protocol could be used for discovery of the local trust network. Then, GKRT (or any other algorithm that relies on global knowledge) could be calculated using the acquired data.

This section presents CloseLook, a computational trust propagation algorithm designed for efficiency and potential for distributed implementation [181].

Extending the Edges of the Trust Network by Similarity Propagation

Similarity propagation can be carried out before the execution of the main CloseLook procedure. During a preprocessing stage, the original trust network is extended by edges that are established as a result of similarity propagation. The strength of these edges is determined by the similarity relation, but also normalized so that it is on the same scale as the strengths of edges in the original trust network.

Algorithm Description

CloseLook was been designed in order to fulfil the following goals:

- the algorithm should be more computationally efficient than GKRT, while achieving comparable quality of results,
- the algorithm should be capable of limiting the amount of computational time used through the use of a scope parameter,
- the algorithm should be capable of using any kind of trust propagation rule, not just transitive propagation, and should be capable of propagating computational distrust as well as trust,
- the algorithm should not be vulnerable to cycles (and therefore should be more resistant to cooperating adversaries),
- the algorithm should be sufficiently generic to solve the computational trust propagation Problems 1, 2 and 3.

The presented algorithm fulfils all the above requirements. The basic idea of the algorithm is to first find the pairs of nodes between which computational trust or distrust can be most likely propagated. The pseudocode of the algorithm is shown in Figure 3.9.

The algorithm is based on a priority queue. In the beginning, the queue is initialized with a set of starting nodes. If the algorithm is used to solve Problem 1 or 2, then the set of starting nodes contains only node A. If the algorithm is used for Problem 3, the set of starting nodes can contain all nodes in V. In each iteration, a node is removed from the queue for processing. The queue is always sorted according to node processing priorities that are calculated on the basis of current knowledge about accumulated trust of a starting node A in the node in the queue using all known paths from A to that node. This ensures that the first pairs of nodes to be processed will be the ones most likely to have a propagated trust edge.

Note that the algorithm is inherently suitable for distributed implementation with incomplete knowledge. As the algorithm proceeds, it requires information about a new trust relation only when it reaches a new node. In such a case, it would be possible to send a message to that node inquiring about its trust and distrust edges. The complete information about the trust and distrust network is not necessary to start the algorithm, and may not be used at all if the algorithm completes its operation without considering all the nodes (see below).

A node can be processed multiple times, but not indefinitely. Node processing is controlled by the *toBeProcessed* method that takes as an argument the starting node from which the considered node has been recently reached. If a node is reached multiple times from the same starting node, then it may be part of a cycle (although not always). Even then, the node may be processed more than once; the number of times that a node can be processed when reached from the same starting node is an algorithm parameter (which has an impact on CloseLook complexity). The condition for processing a node

```
for A ∈ START do
  Queue.push( A, MAX_PRIORITY )
end for
while not Queue.isEmpty and (pCount < scope * (N*size(START))/2) and not stop() do
  u ← Queue.pop
  if not (u.toBeProcessed(u.currentPath.start) and u.currentPath.length ≤ k) then
    continue
  end if
  pCount ← pCount + 1
  p ← u.currentPath
  for e ∈ u.edges do
    p.add(e)
    v ← e.end
    v.updateTotalTrust( p.start, p )
    if e.isDistrust then
      v.updateDistrust( p.start, p )
    end if
    v.priority ← v.totalTrust(p.start) *(0.1)^p.length
    v.updateStatistics( p.start, p, Queue )
    Queue.pushOrUpdate( v, v.priority )
  end for
  u.setProcessedFrom(p.start)
end while
normalizeTrustOrDistrust(Queue)
for A ∈ START do
  for B ∈ V do
    if B.hasBeenProcessedFrom(A) then
      createNewEdge( A, B, B.totalTrust(A), B.distrust(A) )
    end if
  end for
end for
```

Fig. 3.9 Pseudocode of CloseLook

fulfils another purpose: it can limit the lengths of trust propagation paths. A node that is too distant will not be processed further. The parameter k can be used to control the maximum length of trust propagation paths.

Whenever an edge is processed, the counter *pCount* is incremented. This counter is used as a part of the algorithm's stopping condition. The number of processed edges cannot exceed a fraction of the number of all edges in T. The fraction is determined by the *scope* parameter.

The trust between a starting node A and a processed node, B, can be calculated using all available information about the paths from A to B, as well as information about the similarity of A and B using a defined

similarity relation. The actual formula for calculating the priority of a node can depend on the chosen types of trust propagation. Accumulated trust can also be calculated by taking into account similarity propagation separate from transitive propagation. For example, accumulated trust can be a weighted sum of the strengths of computational trust using similarity propagation and using transitive propagation. Based on the results of Guha et al., it would seem advisable to use equal weight for these two types of propagation. Note that the trust established by similarity propagation can be propagated further through transitive propagation. Therefore, the accumulated trust is used further by the algorithm for transitive propagation.

The algorithm updates the current value of accumulated trust from a starting node to a processed node using the method *updateTotalTrust*. Trust is not normalized during transitive propagation, as the new, propagated trust and distrust edges will be created at the end of the algorithm's operation.

The algorithm can apply the transitive propagation of trust and distrust, because nodes are sorted according to the trust of A. Distrust propagation is applied only for one step and creates a propagated distrust relation. Therefore, the propagated distrust value need not be considered further for trust or distrust propagation, and can be remembered in a node attribute (together with the starting node) for further creation of propagated distrust edges.

The priority of a node B in the queue that has been reached recently from a starting node A is calculated by multiplying the currently known total computational trust by $(0.1)^d$, where d is the currently known distance between A and B. This heuristic is similar to the approach used in TidalTrust, where only the shortest paths between nodes are chosen for trust propagation, or to Appleseed, where trust propagation is discounted by the distance between two nodes.

Note that as the algorithm progresses, it will first consider nodes that are closest to a starting node A, and the formula used for node priorities approximates the BFS ordering of nodes. Thus, if B is reached repeatedly from A, it will usually be reached first using the shortest paths, and then using longer paths (or paths of the same length, but smaller propagated trust). As B is reached by the longer paths, it will be moved further down in the priority queue (although the position of other nodes in the queue might change, as well). Note also that the stopping condition based on the *scope* parameter assures that only a part of the possible relationships can be tried. This part will not be divided equally among all starting nodes. Rather, the starting nodes that have strong propagated trust in nodes will be able to create more new trust relations than other starting nodes.

After each processing, a node can be reinserted in the queue. The status of the node is changed to reflect that is has been already reached from a certain starting node. A special rule of the algorithm applies to nodes that have been processed more than once. The rule determines whether such a node can be processed again. By limiting the number of times a node can be processed, it is possible to break cycles. If the number of processing times

is limited to 1, the algorithm will never revisit any node after it has been reached from a particular starting node (although the node may be reached again from another starting node). Thus, only the shortest paths with the highest propagated computational trust will be used to reach any node.

The main loop of the algorithm can also be stopped by an additional condition (computed by the *stop* procedure). The additional stopping condition can be used to change the operation of the algorithm for a solution of Problem 1. Recall that in this computational trust propagation problem, it is necessary to propagate trust or distrust from a starting node A to a specific node B. In our algorithm, when node B is reached, the length of the shortest path from A to B can be recorded (this can be done by the *updateStatistics* function). Then, the stopping condition can limit the length of processed paths to the length of the shortest path from A to B. This kind of operation would be similar to the TidalTrust algorithm; however, TidalTrust also removes all paths that contain cycles or join other paths, and our algorithm takes into account joining paths, while dealing with cycles in a different way.

After the algorithm has completed the propagation loop, it normalizes the resulting trust and distrust values. Normalization is executed globally, rather than locally during the propagation. Normalization depends on the chosen computational trust representation, since the propagated values must be normalized so that they match the limits of computational trust and distrust. Assuming that trust and distrust are represented as reals in the intervals of $[0, 1]$ and $[-1, 0]$, then the normalization function divides the propagated trust by the length of the interval between the maximum and minimum propagated trust values (similarly for distrust).

The final stage of the algorithm is the creation of new trust or distrust edges based on the computed results. For each pair consisting of a starting node and a node that has been reached from the starting node, a new trust edge can be created. This is done by the *createNewEdge* function. The strength of the new edge is equal to the normalized propagated trust or distrust value between the nodes.

3.6.5 Algorithm Evaluation

3.6.5.1 The Complexity of CloseLook

In this section, we discuss the computational complexity of the proposed algorithm and compare it with GKRT. Let us, for the purpose of this section, introduce the notation: n is the number of nodes of the input trust network ($|V|$), while m is the number of edges of the input trust network ($|T|$).

CloseLook uses a priority queue that can be variously implemented. Assume that the queue is implemented using a Fibonacci heap. The number of iterations of the main loop depends on the number of insertions into the queue. In the beginning, there may be up to n start nodes. The number of times that each nodes is inserted into the queue is limited by a constant,

J. This means that there may be $O(Jn) = O(n)$ main loop iterations. During each iteration, there is a queue pop operation, which for a Fibonacci heap has the cost of $O(\lg n)$, and a queue insert operation with the cost $O(1)$. Then, there may be queue updates when the edges attached to the chosen node are processed. Pessimistically, because we have m edges and each is connected with two nodes, which might be processed J times we get: $O(2Jm) = O(m)$ updates($2J$ is constant). In a Fibonacci heap, the amortized cost of an update is $O(1)$. The cost of the trust processing functions of CloseLook (updateTotalTrust, updateDistrust, updateStatistics) is $O(1)$, and these functions are called each time an edge is processed, giving the cost of $O(m)$. Thus, the total pessimistic computational cost of Closelook is $O(Jn(\lg n + 1) + 2Jm + Jm) = O(n \lg n + m)$.

The computational complexity of GKRT is quite high. The cost of a simple implementation is of the order of $O(n^3)$, where n is the number of nodes in the trust network (the cost may be reduced to $O(n^{2.376})$ using advanced matrix multiplication algorithms). Even if we limit the algorithm to a subnetwork within a radius of three, this subset might still be too large in order to efficiently calculate the results (imagine a trust network with average degree of 10. In a radius of 3 within a random node, there will be on average 1000 nodes. The complexity of GKRT is then 10^9). While GKRT has many advantages, due to the fact that it can take into account various methods of propagation, as well as distrust, its computational cost is too high to be used in many practical applications. Also, GKRT attempts to take into account all the data. TidalTrust and Sunny are examples of approaches that limit the amount of information used in trust propagation to reduce the noise induced by many weak edges and paths in the trust network. Despite (or because of) an aggressive filtering of the input data, both algorithms work remarkably well. While TidalTrust and Sunny are algorithms for Problem 1, a similar approach can be used to improve the performance of an algorithm for Problem 2 or 3.

CloseLook is more computationally efficient than GKRT, because it incurs a $O(n \lg n + m)$ computational cost even if its computation is not limited in scope (when similarity propagation is not used. If similarity propagation is used, computational cost may increase depending on the type of similarity relation). For the example trust network with average degree of 10 in a radius of 3 within a random node, this gives a pessimistic complexity (assuming a fully connected network) of the order of 10^6, which is an improvement of 3 orders of magnitude over GKRT. If scope limitation is used, the complexity of Closelook decreases even more.

3.6.5.2 The Correctness of CloseLook

We have evaluated the correctness of the proposed algorithm by comparing it with GKRT. We have used the epinions dataset, in order to better replicate

the experiments described in [57]. Following the approach described there, we have limited the operation of the algorithms to randomly selected sample subsets of the entire trust network. A sample subset was obtained by randomly choosing a starting node A and taking a portion of the trust network reachable from A within a specified radius of $r \in 2, 3$. In the described experiments, computational trust or distrust were propagated from A to all other nodes (effectively reducing Problem 3 to Problem 2). This approach limits the complexity of GKRT (reducing it by a constant factor), because instead of matrix multiplication, a trust vector was multiplied by the combined matrix of trust operators. For our algorithm, we have changed the set of start nodes to include only A.

The samples were chosen so that they would include all trust edges between the nodes within radius $r = 3$ from A. Because of this, the similarity measure of Guha et al. (the co-citation measure) could be calculated on the sample without error (all edges required to calculate the co-citation similarity of any two nodes in the sample were present in the sample). Sample subsets were chosen at random, but in a way that would exclude degenerated cases in which the starting node A would trust very few or very many neighbors. Therefore we have excluded cases where a chosen starting node A had less than 3 or more than 12 direct neighbors in T. Also, we have excluded cases where the subgraph within a radius of 3 from A had less than 50 nodes (as this would imply that the neighbors of A have a very low degree). Excluding such degenerated cases allowed us to run our algorithm on more test samples where computational trust propagation could more easily occur. In the epinions dataset, there are over 1000 nodes that fulfil the described requirements.

Overall, the selected test cases were obtained by taking sample subgraphs from 325 nodes that fulfilled the criteria described above. As in the approach described in [57], we have tested the correctness of our computational trust propagation algorithm by removing an edge originating in the center node A of a sample subgraph, and then running the propagation algorithms starting from A. A different edge could be chosen for the same sample, resulting in more test runs for the same sample. A total of 1339 test cases were selected in such a manner.

The epinions service uses declarative computational trust and distrust, expressed on a scale of $-1, 0, 1$. However, our dataset only had values of $0, 1$; because of this we could not evaluate distrust propagation (which may have decreased the performance of GKRT). The strengths of edges obtained through trust propagation may be rounded to one of these values. This has been the approach adopted by Guha et al.. However, the correctness measure used in this paper has been specified without requiring the rounding of the propagated trust strengths. Such an approach has the advantage that the quality of the propagation algorithm is evaluated directly without considering the additional effect of the rounding procedure.

We have tested several configurations of GKRT. The first group of configurations was implemented as defined in the original paper. The second group

was limited to only transitive propagation. Majority rounding was used for all configurations of the GKRT algorithm. Also, for both variants we have limited the propagation scope to 1 (propagating using two adjacent edges), 2 or 3. The same scope limitation was used for CloseLook; however, CloseLook could additionally limit its scope by considering only a certain number of edges. In our experiments, this number was calculated as a percentage of the edges in the graph G_i.

Figure 3.10 shows the results of comparing the total inclusion measure TIM_λ of GKRT and CloseLook as a function of λ. CloseLook algorithms are denoted by a prefix "CloseLook" on the figure, while the two other numbers are the scope parameters: the number of hops and the percentage of considered edges. GKRT has the prefix "GRKT", while "GRKT_DEF" denotes the original variant, and "GRKT_T" denotes the variant limited to transitive propagation only.

None of the variants of GKRT outperforms the worst variant of CloseLook. Even for low values of λ, CloseLook performs better than GKRT. This is particularly important as very high values of λ (over 400%) are required for high values of TIM. The reason for this is that TIM is a conjunction of two conditions: 100% recall and 100% consistency.

Figure 3.11 shows the sensitivity of CloseLook to the scope parameters (without taking into account the results of similarity propagation. For these experiments, CloseLook was run on the original trust network that did not include edges created by similarity propagation.). Recall that CloseLook uses a conjunction of the two scope parameters in its stopping condition: the

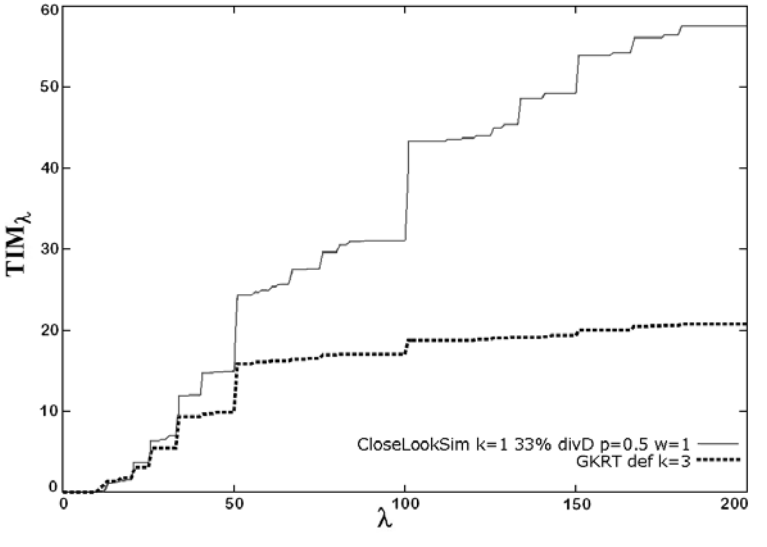


Fig. 3.10 Comparison of GKRT and CloseLook

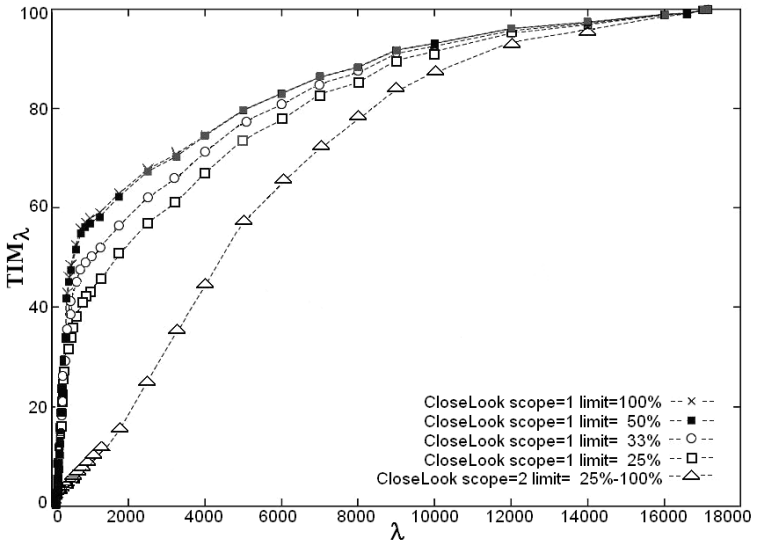


Fig. 3.11 Sensitivity of CloseLook to scope limitation

algorithm stops if the number of hops exceeds the hop limit or if the number of processed edges exceeds the edge limit. It can be seen that in all cases, a lower hop limit gave better results. This is consistent with the observations made by the authors of TidalTrust that shorter propagation paths give better results [54]. Given a fixed hop limit, CloseLook performs better for higher values of the node limit (in our experiments, the edge limit was set to 25%, 50%, 75% or 100% of the size of G_i for any test case). However, this improved performance comes at the cost of additional computation. For values of the edge limit as low as 25%, CloseLook still outperforms GKRT and gives reasonably good results. This implies that for resource-constrained devices, it is still possible to achieve good computational trust propagation using the scope limitations of CloseLook.

We have compared the execution times of GKRT and CloselLook. Recall that the complexity of CloseLook and GKRT increases when similarity propagation needs to be considered. The results confirm our analytical comparison of CloseLook and GKRT complexity. For small graphs, GKRT may execute as fast as CloseLook when both algorithms are limited to transitive propagation. This is due to the fact that GKRT uses very simple data structures (tables), while CloseLook uses a priority queue and object representations of nodes. However, for larger graphs (5000 nodes and above), the execution times of CloseLook are on average about 7.7 times smaller than the execution times of GKRT. The total execution time of CloseLook on the sample graphs of over 5000 nodes was usually less than 1 second (on a fast machine. On a resource-constrained device, execution times may be larger). The difference

between CloseLook and GKRT would have been larger if we had not considered a constant overhead of loading program and data into memory, which is similar for both programs and can be large for large graphs; however, our comparison gives an idea of the practical advantage of using CloseLook.

3.6.6 Conclusions from CloseLook Evaluation

In this section, we have proposed a new computational trust propagation algorithm, called CloseLook. Under the presented classification of trust propagation algorithms, CloseLook is an algorithm for Problem 2 and for Problem 3; however, with some modifications of the stopping condition it can also be applied for Problem 1. The algorithm is based on the principle of limiting the amount of computation (and considered information) by selecting the best paths to propagate trust and by stopping the trust propagation using scope parameters that can limit the number of considered nodes. This property is especially important for resource-constrained devices. CloseLook is also inherently suitable for distributed implementation, because it can operate using only partial information about the trust network.

We have compared CloseLook with GKRT, which is unique among the other trust propagation algorithms described in the literature, because it can consider transitive and similarity propagation, as well as transitive trust and distrust propagation. CloseLook is also capable of considering the same kinds of propagation, but is more efficient than GKRT.

GKRT has been tested using an experiment on a sample trust network that relied on removing edges and attempting to rediscover using the trust propagation algorithm. However, just a single correctness measure - recall - was used in the algorithm's evaluation in the original work of Guha et al.. In this paper, we have proposed three correctness measures: recall, consistency and edge growth. The three measures have further been reduced to two measures: the total inclusion measure which is a conjunction of 100% recall and 100% consistency and λ, which is a measure of edge growth. These two correctness measures form a tradeoff: the total inclusion measure can be seen as an increasing function of λ. However, increasing λ is equivalent to allowing higher edge growth, which increases the risk that the propagation algorithm will make incorrect recommendations. Using the proposed testing method, it is not possible to evaluate the correctness of recommendations made by the trust algorithm for nodes that have before been strangers. Still, limiting edge growth through keeping low values of λ is a sensible precaution.

We have found that the total inclusion measure is a very strict measure of trust propagation algorithms (very high values of λ are required for high values of TIM). However, for all investigated values of λ, CloseLook has outperformed GKRT. This property can be explained by the observation that limiting the information considered by a trust propagation algorithm makes it more robust, as well as reduces the computation time. Therefore, we find

that CloseLook is well suited not only for resource-constrained devices, but for general-purpose trust propagation.

Future work will be directed at a further evaluation of CloseLook. In particular, we plan to compare it with the TidalTrust algorithm in order to evaluate its effectiveness in solving Problem 1.

3.7 Algorithms for Calculating Reputation

The problem of calculating reputation is in many respects similar to the problem of calculating computational trust. The main difference is that in the case of reputation, we are always dealing with reports, and not with recommendations. Interestingly, reports may be weighted by computational credibility trust that is propagated transitively (a common feature of reputation algorithms, such as the Eigentrust algorithm discussed in section 3.6.3.3). In this book, the two problems: of calculating reputation and propagating computational trust, are separated. In this section, we deal solely with the problem of calculating reputation, when a solution to the computational trust propagation problem is given.

Another difference in calculating reputation is that often what is desired is a so-called *global* reputation that takes into account all reports about an agent B that are available. Global reputation is no longer a relational value, such as computational trust. The main reason for this simplification is the desire to produce rankings of agents based on a single value. Recall that solutions to computational trust propagation problems may also produce rankings.

It is possible to completely reduce a reputation system to a trust management system that relies on reports, but does not use the concept of reputation. This can be done in the following manner. Let us denote all available reports in a given context from an agent A about an agent B by Rep_{AB}. The set Rep_{AB} may be transformed into a single value that represents the overall trust of A in B (in the context of the reports). This value may be calculated variously, either as a simple average, or as a moving average that depends on the time ordering of the reports (an extreme would be to simply take the most recent report as the summary value, not taking into consideration the previous information). The summary of Rep_{AB} may then represented as a recommendation. As a result, the trust management system will use only recommendations that can be an input for computational trust propagation algorithms. These algorithms may produce rankings or recommend computational trust values for previously unknown agents, as described in the previous section.

On the other hand, reputation is an intuitive concept that is used by human societies. It is therefore useful to calculate reputation values that may be presented to human users who may make their own trust decisions based on this information.

3.7.1 Simple Global Reputation Algorithms

Consider the set $Rep_B = \bigcup_A Rep_{AB}$ that is the set of all reports available about agent B. Further, consider the set $Rep_B^{'}$ which is the set of most recent reports about B, for example, the reports available within a specific past period – like the last month. Assume that the value of a report is given by the function $val(r)$. The two sets are the basis of calculating simple global reputation values:

The first formula for reputation, $R_1 = \sum_{r \in Rep_B} val(r)$, is a simple sum of all ratings. A refined version of this is a sum of only the most recent ratings, $R_2 = \sum_{r \in Rep_B^{'}} val(r)$. Another possibility is calculating average rate values, $R_3 = R_1/|Rep_B|$ or $R_4 = R_2/|Rep_B^{'}|$.

The use of more recent reports may be motivated by the desire to reduce the noise in reputation calculation, especially if it is possible to assume that the behavior of agents changes infrequently. However, if the reports are unreliable, the average of all ratings is more resilient. If there are only two possible report values (positive and negative), then the average of all ratings also approximates the probability of fair behavior (if the behavior of agents can be modeled probabilistically).

3.7.2 Reputation Algorithms in Internet Auctions

Most online auction sites use a simple feedback-based reputation system [141]. One of the most important tasks of reputation systems is to provide an economical incentive to behave honestly [8] and to reward participants for fair behavior [112]. Typically, parties involved in a transaction mutually post feedback after the transaction is committed. Each transaction can be judged as 'positive', 'neutral', or 'negative'. The reputation of a user is simply the number of distinct partners providing positive feedbacks minus the number of distinct partners providing negative feedback. As pointed out in [104], such simple reputation system suffers from numerous deficiencies, including the subjective nature of feedbacks and the lack of transactional and social contexts. This points to yet another drawback of feedback-based reputation systems: these systems do not account for the psychological motivation of users. Many users refrain from posting a neutral or negative reports in fear of retaliation, thus biasing the system into assigning overestimated reputation scores. This phenomenon is manifested by high asymmetry in feedbacks collected after auctions and, equally importantly, by a high number of auctions with no reports provided. Much of this missing feedback conveys implicit and unvoiced assessments of poor seller's performance which must be included in the computation of seller's reputation in orderto provide an unbiased estimation of the seller's reliability.

This section introduces the concept of an implicit report. An implicit report is a useful, actionable pattern hidden in large amounts of online auction

data. The history of user feedback can be used to discover missing feedback purposely left out and can be included as implicit feedbacks in reputation scoring. We present an efficient and flexible strategy for identifying implicit feedbacks and compare it to a simple majority voting strategy, based on a large volume of transaction data from a real Internet auction. This proposed algorithm to calculate reputation using implicit reports is then evaluated using game-theoretic simulation. The results of conducted experiments clearly indicate a significant impact of using implicit feedbacks in reputation scoring.

3.7.2.1 Existence of Implicit Feedback

A close investigation of the distribution of feedbacks reveals a striking deviation. The examined dataset contains data on a sample group of 10 000 buyers collected over a period of six months. There are 656 376 committed auctions and 890 876 cases mutual feedback. Table 3.6 summarizes data statistics.

Table 3.6 Distribution of feedbacks

	negative	%	neutral	%	positive	%	$\sum$	%
buyer	4318	1%	2877	0.6%	445 723	98.4%	452 918	69%
seller	2558	0.6%	553	0.1%	434 847	99.3%	437 948	66%

Buyers provided 452 918 items feedback, which accounts for 69% of all examined auctions. Note that over 30% of all auctions are not sealed with feedback. Almost all registered feedback is positive (98.4%), with only 1% of negative and 0.6% neutral feedback. Similar characteristics can be observed for feedback provided by sellers, although sellers are slightly less eager to provide feedback in general. Similar results are reported in [142], so we believe that such distribution is quite typical for online auction sites. Table 3.6 presents a grossly optimistic view of the quality of service offered by participants. There are two interesting points to make. First, neutral feedback is missing, the scope for positive feedback ranges from open praise to the acknowledgement of a correct auction (but nothing more), and negative feedback appears only when the quality of service becomes totally unacceptable. Secondly, more than 30% of auctions did not finalize with feedback. In many of these auctions sellers conducted poorly, but the quality of service was either bearable, or the buyer was intimidated and afraid of retaliatory negative feedback. In both cases the reputation of a seller should be affected negatively. We refer to this purposely omitted feedback as implicit feedback that indicate a seller's performance that is unsatisfactory and not deserving praise, yet passable. Listening to this silent, unvoiced feedback makes the reputation estimation more credible. Alas, current reputation systems are not aware of the existence of implicit feedbacks and do not incorporate implicit feedback into reputation scoring.

Not every item of missing feedback should be regarded as an implicit assessment of the user's performance. Feedback might be missing for various reasons, e.g., one of the trading parties might be an inexperienced user who does not know how to post feedback. One simple strategy is to check the history of user's feedback and compute the ratio of user's auctions for which a given user has posted a feedback. If the majority of user's auctions have been sealed with feedback, missing feedback for a given auction might indicate a purposeful omission of feedback, i.e., implicit feedback. We call this strategy the *majority voting strategy*. We also devise a more complex and flexible *cosine strategy* presented next. Let $F(u_i) = \langle f_1, f_2, \ldots, f_n \rangle, f_i \in \{0, 1\}$ be a chronologically ordered list of feedback flags posted by the user u_i, where $f_k = 0$ denotes the fact that the user u_i did not provide feedback for her k-th auction, and $f_k = 1$ denotes the fact that the user u_i explicitly provided feedback for her k-th auction. We arbitrarily assume that the effect of each auction experience (either positive or negative) influences the next two auctions of a given user[5]. $F(u_i)$ can be transformed into an ordered list of trigrams $T(u_i) = \langle t_1, t_2, \ldots, t_{n-2} \rangle$, where $t_i = f_i f_{i+1} f_{i+2}$ is a binary concatenation of feedback flags for the i-th auction with feedback flags for the consecutive two auctions. There are $2^3 = 8$ possible trigrams represented by binary numbers ranging from 000 (three consecutive auctions do not have feedback) to 111 (three consecutive auctions have feedback). Thus, $T(u_i)$ can be represented as a vector $\bar{T}(u_i) = \left[t_i^0, \ldots, t_i^7\right]$, where t_i^n is the number of occurrences of the n-th trigram in $T(u_i)$. We perceive $\bar{T}(u_i)$ as a condensed representation of feedback habits of the user u_i. Having transformed the original history of user feedback into an 8-dimensional vector we can compare this vector to a template vector representing a user who almost never provides feedback for her auctions (in our experiments we have used the template vector $\bar{T}(0) = [1, 0.1, 0.1, 0.01, 0.1, 0.01, 0.01, 0]$, where three consecutive auctions without feedback have the weight 1, two missing cases of feedback have the weight 0.1, and one missing case of feedback has the weight 0.01). Let k-th auction of the user u_i does not have feedback. First, we build $F(u_i) = \langle f_1, f_2, \ldots, f_k \rangle$, which is transformed into $T(u_i) = \langle t_1, t_2, \ldots, t_{k-2} \rangle$, and the resulting list $T(u_i)$ is transformed into the vector $\bar{T}(u_i)$. Next, we compute the Ochini coefficient (the cosine similarity function) between $\bar{T}(u_i)$ and $\bar{T}(0)$ as follows

$$Ochini(\bar{T}(u_i), \bar{T}(0)) = \frac{\sum_{k=0}^{7} t_i^k * t_0^k}{\sqrt{\sum_{k=0}^{7} (t_i^k)^2 * \sum_{k=0}^{7} (t_0^k)^2}}.$$

[5] The authors acknowledge that the choice of three consecutive auctions as the range of psychological influence of an auction outcome is arbitrary and the correctness of this assumption remains open for discussion.

If $Ochini(\bar{T}(u_i), \bar{T}(0)) < \beta$, where β is a user-defined threshold, we conclude that the two vectors are similar and the omission of feedback should not be regarded as implicit feedback.

Example. Let us assume a user u with the following list of feedback flags: $F(u) = \langle 0, 1, 0, 1, 1, 0 \rangle$. The user u participated in six auctions and did not provide feedback for three of them. We want to know if the last missing item of feedback is a purposeful omission. First, the list of feedback flags $F(u)$ is transformed into a list of trigrams $T(u) = \langle 010, 101, 011, 110 \rangle$. Next, the list of trigrams is transformed into a compact vector representation $\bar{T}(u) = [0, 0, 1, 1, 0, 1, 1, 0]$. The final result is $Ochini(\bar{T}(u), \bar{T}(0)) = 0.09$. After a certain period of time the user u participates in more auctions and gains experience. Let us assume that, after a while, the list of feedback flags for the user u is $F(u) = \langle 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0 \rangle$. We want to decide on the last missing case of feedback as being an implicit feedback. The list of trigrams is $T(u) = \langle 010, 101, 011, 110, 101, 011, 111, 110, 101, 011, 110 \rangle$ and the vector representation is $\bar{T}(u) = [0, 0, 1, 3, 0, 3, 3, 1]$. Now the computation of the Ochini coefficient yields $Ochini(\bar{T}(u), \bar{T}(0)) = 0.035$. As can be seen, this procedure is flexible and allows for temporal changes in feedback habits.

To prove the existence of implicit feedback we begin by investigating the distribution of the number of missing cases of feedbacks per user (in this experiment we include only buyers). The results of the experiment are depicted in Figure 3.12. Interestingly, there are only a few buyers with more than 20 missing feedback. This might indicate that most of the missing feedback is in fact purposeful omissions, thus turning items of missing feedback into implicit feedbacks. When we have constrained our search to buyers who had participated in at least 10 auctions, the average percentage of missing feedback dropped to 11.6%, which indicates that experienced users are even less likely to omit feedback.

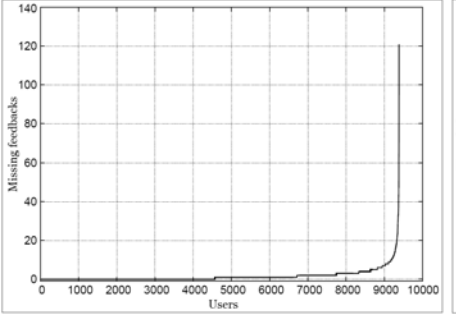


Fig. 3.12 Missing feedback distribution

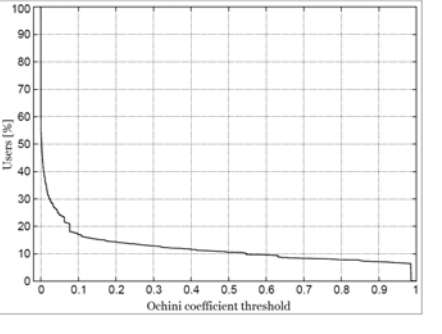


Fig. 3.13 Ochini coefficient selectivity

Figure 3.13 presents user selectivity depending on the value of the Ochini coefficient. Recall that the Ochini coefficient represents the similarity between a given user's feedback vector and the template vector of a hypothetical 'I-don't-do-feedback' user, with the values closer to 1 representing high similarity and the values closer to 0 representing high dissimilarity. The figure presents the percentage of users who would be considered as generally not providing feedbacks, given the value of the Ochini coefficient threshold. For reasonable values of the Ochini coefficient threshold (i.e., 0.5 and above) less than 10% of buyers are regarded as reluctant to provide feedback, which means that their missing feedback would not be considered as implicit feedback. Again, this result proves that for the majority of buyers missing feedback is an important, yet unvoiced, assessment of a business partner's performance.

3.7.2.2 Effectiveness of Using Implicit Feedback

To evaluate a reputation system it is necessary to find out how this system affects the behavior of users and the outcome of user transactions. Ideally, we would like to know whether a reputation system enables trust: all honest users should trust other honest users and should be treated fairly by other honest users. On the other hand, all dishonest users should not be trusted and therefore should not participate in transactions. In this section we compare a simple reputation algorithm, such as the one used by eBay, to a more complex algorithm that uses implicit feedback. Section 3.7.2.2 presents the design of the simulator of online auctions. The results of conducted simulations are reported in Sect. 3.7.2.2.

The Simulator

Prior to starting the simulation we had to make a decision about a sufficiently realistic, yet not too complex model of the auctioning system, of user behavior, and of the reputation system. We choose to simulate the reputation system almost totally faithfully, the only simplification is that we use only positive and negative feedback. The behavior of a user is also realistic: the user takes into account reputation when choosing a business partner. The user also decides whether she wishes to report or not, depending on the type of report. Users can cheat in reports, as well as in transactions. Users may also use transaction strategies that depend on the history of their individual interactions with other participants.

The auctioning system, on the other hand, has been simplified. We reflect that the simulation of the entire auction process is unnecessary. Rather, we simulate the selection of users using a random choice of a set of potential sellers. The choosing user (i.e., the buyer) selects one of the sellers from among users with the highest reputation in the set. The auction itself has also been simplified. We use a popular game-theoretic model of an auction, namely, the iterated Prisoner's Dilemma [7].

In the simulator, a number of agents that represent users interact with each other. Each interaction represents an auction between a seller and a buyer. The reputation system is maintained by a reputation server that is also used to summarize the outcomes of agent interactions. Each agent is characterized by the following parameters: r^+, the probability that an agent will send a report if it is positive, r^-, the probability that an agent will send a report if it is not positive, the chosen game strategy, the reputation threshold ρ_{min} that is used by some strategies, and the probability of cheating c, that is used by some strategies. We can specify the number of agents and every agent can have distinct parameters. However, we usually partition all agents into two sets that have the same parameters, called the honest and dishonest agents.

The two game strategies used in the simulations are: to cheat with the probability c or to play Tit-for-Tat with a reputation threshold ρ_{min}. Tit-for-Tat is a famous strategy for the iterated Prisoner's Dilemma game. This strategy works simply by repeating the move made by the other agent in the previous encounter. If two agents meet for the first time, the classic Tit-for-Tat strategy forces the agents to cooperate, thus allowing the agents to start an unending pattern of honest transactions. We modify Tit-for-Tat to use a reputation threshold: if two agents meet for the first time and the second agents' reputation is below ρ_{min}, the first agent defects.

The server computes reputation scores using available feedback and using any implemented algorithm. The results of the simulation include: reputation scores of individual agents and the total payoffs (from all auctions) of every agent. The payoffs are affected by the way the reputation system works. For example, if agents post very little feedback, reputation scores will be generally random, and the payoffs of good agents would drop. The simulator allows us to check whether the implemented reputation algorithm is effective. To verify the concept of implicit feedback, we simulate the behavior of a simple reputation algorithm that uses implicit feedback.

Consider a user u with the history of n auctions. Let us assume that only $m \leq n$ of these auctions have a feedback. Out of these m feedback m^+ is positive feedback, while $m^- = m - m^+$ is all other feedback. Thus, $m^+ \leq m \leq n$. The reputation ρ_u of the user u will be calculated as follows

$$\rho_u = \frac{m^+}{\alpha * m^- + m},$$

where $0 \leq \alpha \leq 1$. Thus, if $\alpha = 0$, the above reputation score becomes a simple ratio of the number of examples of positive feedback received by the user u. In a case where the user u has had no auctions, the above formula is undefined. In such a case we set the reputation ρ_u to an initial value, ρ_0. To be precise, in our simulations we use a slightly more complex version of the above algorithm. Since agents in the simulator choose whom they want to interact with on the basis of reputation scores, it is necessary to avoid the possibility that reputation would drop suddenly to a low level. This can happen in the initial phase of the simulation, when the reputation

score has not yet stabilized (initially, a single negative item of feedback could decrease the initial reputation by a large degree). Therefore, we use a moving average to smooth reputation changes. The smoothed reputation $\rho_u^{ma}(t) = 0.5\rho_u^{ma}(t-1) + \rho_u(t)$, where t is time, and $\rho_u^{ma}(0) = \rho_0$ - the smoothed reputation is initialized by the initial reputation value. Note that over time, the estimate converges to the formula for ρ_u (since the impact of the initial reputation decreases exponentially).

Evalutation by Simulation

We have tested the algorithm described above using the following simulation scenario. First, we have divided all 300 agents into two sets, the *good agents* and the *bad agents*. Good agents were 66% of all agents, the remaining agents were bad agents. A good agent used the Tit-for-Tat strategy with the reputation threshold of $\rho_{min} = 0.5$. A bad agent used a strategy of random cheating with probability $c = 0.6$. All agents had the same behavior with respect to feedback. This behavior was a further parameter of the simulation scenarios. We used two posting behaviors: *perfect feedback*, where all agents always posted feedback truthfully, and *poor feedback*, where if the feedback was positive, an agent would post it with probability $r^+ = 0.66$, and if the feedback was not positive, an agent would post it with probability $r^- = 0.05$. All items of feedback were always true, if they were sent. The parameters of the poor feedback were derived from the analysis of traces obtained from the real-world data. In all simulations, 40 000 auctions were simulated between the agents.

Together, there are three significant simulation scenarios: perfect reports with reputation calculated using a simple ratio of positive feedback (a reputation algorithm like described in the previous section, only with $\alpha = 0$); poor reports with a simple ratio; and poor reports with the reputation algorithm that uses implicit feedback, with different settings for α.

All experiments were conducted using the Monte-Carlo method. We present average results from 10 simulation runs, together with 95% confidence intervals of results. The outcomes of the experiments were the payoffs of every agent. We evaluate the effectiveness of a reputation system using the following criteria: the average payoff of a good agent, the average payoff of a bad agent, and the Gini coefficient of the payoffs of the good agents. The last criterion was introduced as a way of evaluating the effectiveness of the reputation system in providing fairness of the treatment of good agents.

The results of the simulations are summarized in Table 3.7. The first row in the table corresponds to the perfect feedback scenario, where all agents always post truthful feedback, and reputation is calculated using a simple ratio of positive feedback. In this idealized scenario, the average payoff of good agents is the highest, at 101.76 (the values of payoffs for a single auction were derived from the payoff table of the Prisoner's dilemma game). The average payoff of a bad agent is much lower, at 22.66. This indicates that the

Table 3.7 Impact of implicit feedback reputation algorithm on agent payoffs and justice

Scenario	AVGP+[a]	AVGP-[b]	GC+[c]	GCI[d]	AGPCI[e]
Perfect reports	101.76	22.66	0.70	0.63–0.76	100–103,5
Poor reports, $\alpha = 0$	96.45	54.20	0.51	0.45–0.58	93.6–99.3
Poor reports, $\alpha = 0.05$	99.08	23.03	0.75	0.66–0.85	96.1–102
Poor reports, $\alpha = 0.1$	100.40	23.52	0.67	0.58–0.76	98.8–101.9
Poor reports, $\alpha = 0.2$	100.74	22.64	0.74	0.66–0.82	98–103.4

[a] Average payoff of a good agent
[b] Average payoff of a bad agent
[c] Gini coefficient of good agents
[d] Gini confidence interval (95%)
[e] Average good payoff confidence interval (95%)

reputation mechanism is working because cheating agents get punished by lower reputation. In these simulations, the final reputation value of bad agents was almost always 0. The Gini coefficient at about 0.7 will be treated as a reference level for further experiments and values of the Gini coefficient above this level will be considered unacceptable. The 95% confidence intervals for both the Gini coefficient and the average payoff are quite narrow. The second row of the Table 3.7 shows the results of the second simulation scenario. In this scenario, agents provided feedback realistically, and the effect of this is an increase in the payoff of bad agents by almost 150%. In many simulations, bad agents managed to keep a high reputation value, leading the good agents to trust them. This enabled bad agents to cheat more good agents. As a result, the average payoff of good agents also decreased significantly. This decrease is also visible in the confidence interval of payoffs of good agents.

Further rows of the table show the impact of using implicit feedback. The rows correspond to using the reputation algorithm described in the previous section with different values of α. For all considered values of α, the payoffs of bad agents dropped sharply, almost to the level achieved when agents reported perfectly. This is the main argument for using implicit feedback: as our simulations indicate, the use of implicit feedback is efficient in preventing cheating. The payoffs of good agents also increased to a varying degree, but for all values of α, the average payoff of a good agent was higher than when a simple ratio of positive feedback was used as the reputation algorithm. On the basis of the performed experiments, it seems that the value of $\alpha = 0.1$ gave the best results. For $\alpha = 0.2$, the average payoffs of the good agents were higher, and the average payoffs of bad agents were lower than for $\alpha = 0.1$. However, the average Gini coefficient was also higher. The reason for this may be that in the simulations, good agents sent positive feedback randomly with a probability of 66%. It was possible that a good agent would repeatedly get no positive feedback for her cooperation with another good agent. This could result in decreasing the reputation of the good agent, especially for

higher values of α. The poor performance of $\alpha = 0.05$ can be explained by the fact that with such a low setting of α, the reputation of bad agents did not decrease quickly enough. While our simulations do not allow to choose the value of α that would be applicable in a real-world scenario, they are sufficient to indicate that there should exist an optimal value of α that is neither too high nor too low.

3.7.3 Design of P2P Reputation Systems

Now let us turn to the design of P2P reputation systems. We distinguish four main aspects of every reputation system: types of used proofs, aggregation scope, computed value and resilience to adversaries.

Type of used proofs. Proofs given from peers can be divided into two distinct groups: reports and observations. Reports are the feedback from peers after their own transactions. Observations are information about the transaction from third party peers (observers). Systems like NICE [88], P2PREP [30] or proposed by [148] are based on reports. Proofs can evaluate several criteria like transaction completion, provided transfer, or quality of service. [170] use context-specific ratings which can integrate several criteria (for example, a peer seeks the provider which has high download speed and good quality of audio files). In PeerTrust [184], all types of proofs are used, an proofs are weighted by the credibility of the peer. Also, the context factor is taken into consideration. In addition to PeerTrust, Xiong et al proposed TrustGuard [155] in which they claim that only secure proofs can be used. Their TrustGuard credibility filter is based on the similarity of experience between two peers.

Aggregation scope. *Full aggregation* systems consider proofs from all peers, concerning both about own and third-party iterations. Such aggregation is applied in Eigentrust [69], Gossiptrust [194] and PET [93] systems. This approach has a high accuracy of rating due to the use of full system information, but unfortunately it involves a high computational cost.

Selective aggregation is applied when the performance of the system is more important than accuracy. Systems like [195, 152, 30, 32, 148, 88, 193, 99] gather the information only from a subset of peers (in most cases from their neighborhood).

Computed value. *Global* reputation systems [69, 194, 195, 152] are based on the opinions from the whole peer population. Reputation represents an objective evaluation of the trustworthiness of a peer in the system.

Personalized trust measures [93, 30, 32, 148, 88, 193, 99] are more often used in open peer to peer environments due to problems of obtaining global system information. Such measures are subjective and personalized and can have different values for each asking peer.

System resilience. To prevent peers from cheating, TM systems can use cryptographic methods like Public Key techniques [184, 148, 151, 191].

In [148], every proof provided by peers is signed. Strong authentication of peers is also used in TrustMe [151] by Singh et al to protect the system from adversaries. [191] assumed the use of PKI for naming and authentication in their system in order to detect unreliable or malicious peers.

More sophisticated attacks can be deflected by special algorithms or system designs. [69] deals with malicious collectives in the basic version of EigenTrust by adding a probability of crawling to a peer that has been recommended a-priori (is pre-trusted). In the secure version of EigenTrust, authors prevent peers from manipulating their own trust values t_i by the assuming that the current trust value of host i should be computed by score managers.

Dropping of negative proofs is a challenge in R-CHAIN[99]. To prevent the Sybil attack, R-Chain can enforce some proof of work for peers joining the system.

[148] protect their TM system against denial of service attacks. They add an extra round before the response in which querying peers receive a puzzle scheme. A querying peer decides in which file versions he is genuinely interested, solves the puzzle and sends the proof-of-work back.

Chapter 4
Fairness Management

It is in justice that the ordering of society is centered.
Aristotle

By Fairness Management we refer to a wide variety of techniques that aim to improve the procedural or distributional fairness of ODS. This chapter begins with a consideration of Fairness Management for resource sharing in an ODS, and presents various application areas where such a problem needs to be solved. The discussion of these application areas will be accompanied by case studies that present various Fairness Management techniques. Most of these techniques can be described using the concepts of the theory of fairness introduced in section 2.3. The case studies have been selected so that they demonstrate various kinds of fair distribution problems. The first example of a fair scheduling in grids is a decentralized problem, in which a solution cannot be imposed on the participants. On the other hand, in the second example of a network dimensioning problem, the fair solution can be implemented by a central control – network management. The two cases differ also in other respects; the first problem has an extremely large solution space and a heuristic method is presented for finding equitable solutions, while the second problem is formulated as a Mixed Integer Linear Optimization problem and solved by a CPLEX solver. As a third example, we consider fairness in Peer-to-Peer systems that are even more decentralized than grids.

In section 4.2, we turn to procedural fairness, another important type of fairness management. ODS are a particularly challenging area for procedural fairness management, as their openness and lack of centralized control makes it easy for adversaries to work against the fairness management system. In these respects, procedural fairness management must deal with problems similar to TM techniques. As a matter of fact, it is possible to directly apply Trust Management solutions to the problem of procedural fairness management. Yet, in section 4.2 we begin with the discussion of an approach that uses cryptography to directly guarantee procedural fairness in Peer-to-Peer Massive Multiplayer games. We then turn to distributed priority queues and

A. Wierzbicki: Trust & Fairness in Open, Distributed Sys., SCI 298, pp. 145–224.
springerlink.com

agreement protocols that can be important primitives in Procedural Fairness management.

Following the discussion of various Fairness Management techniques, the chapter describes how it is possible to use Trust Management for distributed, emergent Fairness Management. In section 4.3, we discuss the mechanism of fairness emergence due to trust management, spending some time on a consideration of the conditions required for this phenomenon to occur. The existence of fairness emergence is good news for ODS designers, as it shows that using even very simple Trust Management techniques it is possible to improve distributional fairness as defined by the theory of equitable optimality introduced in section 2.3.

4.1 Distributed Resource Sharing

4.1.1 Grids

Computational Grids [50] are distributed supercomputers of a very large scale. With the growing level of complexity of models used in many areas of modern science on one hand, and the growing volume of experimental data to analyze on the other hand, the access to computational power becomes a key apparatus [21] in areas as diverse as molecular biology, particle physics, physical chemistry, or civil engineering. Computational grid may become a convenient tool that provides the enormous computing power required by such projects. Computational grids have been extensively studied since the end of the nineties. Nowadays it seems that most engineering problems either have been resolved already, or will be resolved soon with the advancement of the middleware, such as the Globus Project [49]. One of the most important remaining challenges is rather of economical, or even psychological nature. The Grid, by its definition [48], is inherently distributed, as it combines resources under different administrative domains. Consequently, certain rules of collaboration must exist, which specify how users from one domain access resources belonging to others. Providers must have some motivation to share their resources, expressed either by earning money for each CPU hour donated, by barter-trading the access, or by formal bilateral agreements between institutions.

The problem of fairness management in grids can be approached differently than in P2P systems. Grids can have some measure of central control, or at least a central authority that can suggest an equitably optimal solution, even if it is not capable of enforcing it. Fairness management in grids can be reduced to a scheduling problem. Since all agents use resources provided by various grid members, the problem is how to schedule the tasks of all users on each grid machine. This problem can be made more concrete by assuming a specific model of the grid. Such a model needs to specify, for example,

whether tasks are infinitely divisible, or not, and whether or not certain grid machines are dedicated to certain types of tasks.

The mainstream of the current research on scheduling and resource management [46] concerns systems in which the performance of all tasks is optimized. Usually, a common metric, such as the makespan, or the sum of completion times is optimized and thus all the jobs are treated in a more or less equal manner.

In the context of Grid computing, *multi-criteria* approaches may be used. Different criteria usually express performance of different jobs [105]. A scheduling algorithm is expected to deliver Pareto-optimal solutions. Further restrictions on Pareto-optimal solutions should be imposed in order to achieve equitably optimal solutions.

Grid economic approaches [23, 183, 76] analyze the problem of grid resource management by means of market economy. Each resource has a (monetary) cost for its usage. Each user has a budget to spend for executing his/her jobs. The problem is that in highly heterogeneous settings the perfect competition assumption, stating that no single participant is able to influence the market price, is hardly fulfilled. Real-world grids, however, are expected to be heterogeneous [56]. Solutions that solve the problem of scheduling in heterogeneous systems directly, without relying on free-market assumptions, are therefore desirable.

There were also some applications of game theory to the problem of grid resource management. [168] focuses on maintaining good relationships of a node with its neighbors by accepting neighbors' jobs to be executed on the node and therefore increasing the probability that the node's jobs will be accepted by its neighbors in the future. [87] proposes a model where individual clusters (placed in e.g. different departments of an university) are visible as one site in the grid. The model assumes that a job has been already accepted for execution by the site. [87] studies which cluster from that site should eventually execute the job. There was also some previous work where the infrastructure was considered a common property and there was selfishness between individual jobs [5],[98].

An approach that combines game theory and equitable optimization is presented in [145]. The authors introduce an algorithm (called Equitable Walk) that can discover equitably optimal solutions to the global scheduling problem. These solutions can then be proposed by a centralized scheduler to the grid members. Since the solutions are equitably optimal, it can be assumed that fair agents would accept such a schedule. However, malicious agents may still exploit the system by locally modifying the equitably optimal schedules for their own advantage. Still, if the number of malicious agents is small, the existence of equitably optimal schedules should provide an incentive for fair agents to participate in the system. The behavior of unfair agents could be controlled by a trust management system.

4.1.1.1 Dedicated Grids

A special case of a grid could be a system that is composed not only of computing devices (processors), but also of special-purpose experimental equipment like sophisticated displays, microscopes, or DNA sequencers. Any computation on the grid may involve ordinary computers, as well as the special devices. This means that the computation must be decomposed into jobs that can only be run on one type of device in the grid. This grid model is referred to as a dedicated grid model, since certain processors are dedicated for certain jobs. The dedicated model can be useful even when there are no special processors in the grid: computers could be dedicated to running legacy software, like Fortran code.

Therefore, a computational grid logically interconnects several processing units such as clusters, supercomputers, but also pieces of specialized equipment (Figure 4.1). The actual physical network, consisting of high-performance network links, is out of the scope of this section. Users are grouped in organizations, such as laboratories or faculties. Each organization owns some (without loss of generality, we shall assume that it owns just one) of the processors, which is the organization's contribution to the grid. By contributing, the organization expects that its users will be granted access to other processors in a fair manner. Each organization is concerned only with the performance (measured as the sum of completion times) of the jobs produced by its members. Processors have their local schedulers, which order jobs to be computed according to some criteria. A centralized grid scheduler helps to coordinate local schedulers. However, the organizations are independent, i.e. a local scheduler is not forced to follow the advice given by the grid scheduler. This model is a typical example of an ODS. It also shows that fairness expectations are made by the organizations who are agents in this ODS.

In this section we address two issues concerned with such an architecture. Firstly, what should be the properties of the schedules proposed by the grid scheduler to be acceptable by local schedulers. Secondly, how can the grid scheduler produce such schedules. The results presented in this section are based on [145]. The reader is referred to this publication for proofs of cited facts.

4.1.1.2 Notation and Preliminary Definitions

By $\mathcal{O} = \{O_1, \ldots, O_m\}$ we denote the set of independent organizations forming the grid. Each organization O_i owns a processor P_i. Processors are *dedicated*. Therefore, each job in the system must be executed on a specific processor.

$J^k_{i,q}$ is kth job which must be executed on processor P_q and which is produced (and owned) by organization O_i. Index k is used only to distinguish between jobs on a processor and does not imply the arrival or execution

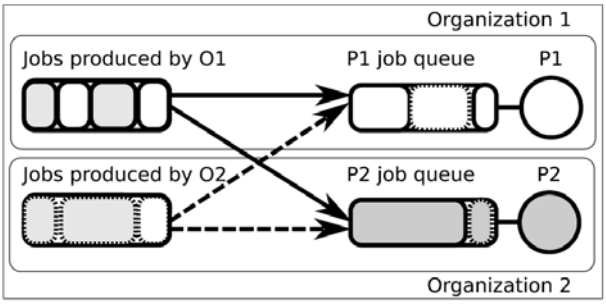


Fig. 4.1 A grid composed of two organizations and two dedicated processors. P_1 processes white jobs, P_2 processes gray jobs. O_1 produced 7 jobs (plotted in continuous lines), 4 of them are not yet sent, two are waiting in P_1's queue and one is waiting in P_2's queue. O_2 produced 5 jobs (plotted in dotted lines).

order. By $\mathcal{J}_{i,q}$ we denote the set of all jobs produced by O_i which must be executed on P_q. $n_{i,q} = |\mathcal{J}_{i,q}|$ is the number of such jobs.

We will use the standard game-theoretic notation of $-i$ to denote the set containing everything but the element i, so $\mathcal{J}_{-i,q} = \left(\bigcup_j \mathcal{J}_{j,q}\right) - \mathcal{J}_{i,q}$. Similarly, by $\cdot$ we denote the set containing all possibilities, e.g. $\mathcal{J}_{i,\cdot}$ is the set of jobs that are produced by organization O_i: $\mathcal{J}_{i,\cdot} = \bigcup_j J_{j,q}$.

From processor's P_q point of view, jobs produced by the processor's organization $\mathcal{J}_{q,q}$ are called the *local jobs*. Remaining jobs $\mathcal{J}_{-q,q}$ to be executed on this processor are called the *foreign jobs*. For organization O_i, *remote jobs* $\mathcal{J}_{i,-i}$ are the jobs produced by this organization which are to be executed on non-local processors.

A scheduler is an application which assigns start times to jobs. A scheduling problem is the problem of how to assign such start times (formal definitions follow). A scheduling problem is considered *off-line* if all the jobs are known before the scheduling starts. Here, we also consider that all the jobs are ready to be executed (there are no *release dates*). A *clairvoyant* scheduler knows the size $p^k_{i,q}$ of each job $J^k_{i,q}$. There is *no preemption* if the job must be executed completely and cannot be interrupted after a processor has started to execute it. A processor can be shared by the jobs assigned to it in many ways. In *time sharing*, at any moment, a processor executes only one job.

By $C^k_{i,q}$ we denote the completion (finish) time of a job $J^k_{i,q}$. For an organization O_i, we may compute the sum of completion times as $C_i = \sum_q \sum_k C^k_{i,q}$ and the maximum completion time (makespan) as $C_{\max i} = \max_{k,q} C^k_{i,q}$. In the classic multiprocessor scheduling problem, the optimization of ΣC_i results in schedules better for users, whereas the optimization of the makespan $C_{\max}$ produces schedules which utilize the machine better [43].

For processor P_q, a *(list) schedule* is a permutation $\pi : \mathcal{J}_{\cdot,q} \to (J^{\pi(1)}_{\cdot,q}, J^{\pi(2)}_{\cdot,q}, \ldots, J^{\pi(n_{\cdot,q})}_{\cdot,q})$ of jobs $\mathcal{J}_{\cdot,q}$. A *scheduler* is an application which produces schedules, given the sets of jobs assigned to each processor. A non-preempting,

time-sharing processor executing a schedule π, firstly executes the first job $J_{\cdot,q}^{\pi(1)}$, then, when the jobs finishes, executes the second one $J_{\cdot,q}^{\pi(2)}$ and so on. A Shortest Processing Time (SPT) schedule π_{SPT} is a schedule which orders the jobs $\mathcal{J}_{\cdot,q}$ according to non-decreasing sizes of jobs, i.e. $p_{i,q}^{\pi_{SPT}(k)} \leq p_{i,q}^{\pi_{SPT}(k+1)}$. If there is one processor and one organization, SPT schedule is optimal regarding the sum of completion times of jobs [15].

4.1.1.3 Problem Statement

We consider off-line, clairvoyant scheduling with no preemption on time-sharing processors. Each organization O_i is concerned with the sum of completion times C_i of the jobs $\mathcal{J}_{i,\cdot}$ which have been produced by its members. Organization O_i does not care about the performance of other organizations. However, as the processors are dedicated, organizations must submit jobs also to non-local processors. As a result, C_i depends heavily on the performance of jobs executed on processors other than the organization's local processor, which in turn are controlled by other organizations.

The scheduling problem considered in this section is the *fair* ordering of jobs $\mathcal{J}_{\cdot,q}$ on each processor P_q which minimizes the sum of completion times C_i of all the organizations. We follow the axiomatic theory of equitable optimality, as stated in Section 2.3.2.

Informally, equity guarantees that the resulting sum of completion times would be fair to every organization. However, if each organization is able to improve its objective by introducing some local modifications in the schedule, the resulting system must be analyzed by *game theoretic* approaches.

The following general property applies in both equitable optimization and in game-theoretic approach:

Proposition 4.1. *For a processor P_q, a schedule which orders jobs $\mathcal{J}_{i,q}$ originating from a organization O_i not in shortest processing time (SPT) order is Pareto-dominated by a SPT schedule for those jobs.*

Note that this proposition does not apply for jobs belonging to different organizations. Given two jobs with different owners $J_{i,q}^k$ and $J_{j,q}^l$, $J_{i,q}^k$ executed before $J_{j,q}^l$, swapping them would increase C_i, at the same time decreasing C_j. Consequently, neither solution would Pareto-dominate.

The consequence of the proposition presented above is that the number of reasonable (Pareto-efficient) schedules is reduced considerably. However, their number is still very large. Let us assume that there are only two organizations O_1 and O_2. Consider processor P_1 with $n_{1,1}$ local jobs and $n_{2,1}$ foreign jobs. A multiset (called also a bag) is a set which may contain multiple copies of an element. The schedule can be fully described by a multiset of size $n_{2,1}$, with elements $\{1, \ldots, n_{1,1}+1\}$. The idea is that each member of the multiset specifies a placement for one foreign job. In order to construct a schedule from the multiset, the local jobs are ordered according to SPT. Then, the

shortest foreign job is assigned the smallest element from the multiset, which specifies its placement (1 – the job is placed before the shortest local job, 2 – before the second shortest local job, ..., $n_{1,1}+1$ – after the longest local job). This element (actually, one of the copies of the element) is removed from the multiset. The algorithm proceeds with the rest of foreign jobs (considering them in SPT order). Consequently, the number of SPT schedules for one processor is equal to the number of multisets of size $n_{2,1}$ with elements $\{1,\ldots,n_{1,1}+1\}$, which in turn is equal to the number of combinations with repetition of $(n_{1,1}+1)$ objects from which $n_{2,1}$ objects are chosen:

$$\binom{n_{1,1}+n_{2,1}}{n_{2,1}} = \frac{(n_{1,1}+n_{2,1})!}{n_{1,1}!n_{2,1}!} \in O(n_{1,1}^{n_{2,1}}).$$

Note that although not necessarily all SPT schedules are Pareto-optimal, in the worst case the number of non-dominated schedules for a single processor is exponential with the number of foreign jobs [4]. A complete schedule on the grid level specifies schedules for each processor. In a two processor case, each of $O(n_{1,1}^{n_{2,1}})$ schedules of the first processor is matched with $O(n_{2,2}^{n_{1,2}})$ schedules of the second processor. Most of the resulting combinations are Pareto-dominated. However, at least $O(n_{1,1}^{n_{2,1}})$ Pareto-optimal schedules remain. Thus, the number of Pareto-optimal grid schedules is also exponential.

4.1.1.4 Equitable Walk (EW)

In this section we assume that organizations cannot control the schedule on their processors. The problem considered is how to construct a grid schedule (consisting of schedules for each processor) which would treat all the participating organizations fairly, at the same time as being efficient. This problem is, in fact, multicriteria minimization of the sum of completion times C_i of different organizations. In this section we will firstly characterize some general properties of *equitable multicriteria optimization*, and then propose an heuristic approach which produces equitable solutions of the multicriteria minimization – a local search method called Equitable Walk.

EW is a heuristics which produces a number of grid schedules by iterative modifications of the initial SPT schedule in order to improve the outcome of the organization disfavored by the SPT schedule. The resulting schedules are possibly equitable: the algorithm may produce non-equitable schedules and not all possible equitable schedules may be produced.

The algorithm modifies the schedules by *switching* the order of two jobs executed one after another on the same processor. Let us assume that job $J_{i,q}^k$ is executed before $J_{j,q}^l$. The *deterioration* from a particular switch can be defined as the difference between the decrease of C_j and the increase of C_i. For instance, if $p_{i,q}^k = 3$ and $p_{j,q}^l = 4$, the switch of order of those two jobs

will reduce C_j by 3 and increase C_i by 4. The deterioration of that particular switch is thus 1.

Corollary 4.1. *A schedule which orders jobs on all processors according to SPT (regardless of their owners) is equitable.*

This is a direct consequence of Theorem 1. Such schedule (denoted SPT) is optimal with regard to the sum of completion times of all jobs on all processors ($\sum_i \sum_q \sum_k C^k_{i,q}$). Therefore, it is also the minimal sum of sum of completion times of respective organizations ($\sum_i C_i$). Hence, SPT is Pareto-optimal solution of the aggregated equitable optimization problem, as it minimizes the sum of all criteria.

The algorithm starts with ordering jobs $\mathcal{J}_{\cdot,q}$ on every processor P_q according to SPT. Then, the organization O_i with the largest C_i is selected. For each O_i's remote job $J^k_{i,q}$, the algorithm tentatively advances the job by switching $J^k_{i,q}$ with the job executed immediately before. Similarly, each foreign job $J^k_{\cdot,i}$ executed on O_i's local processor P_i which is followed by a local job $J^l_{i,i}$ is tentatively delayed by switching $J^k_{\cdot,i}$ with $J^l_{i,i}$. From all the tentative moves performed, the one which results in the smallest deterioration is actually performed. Then, the grid schedule is appended to the list of results.

The algorithm iterates such moves until either C_i is no longer the largest completion time, or no further improvement is possible. In the first case, the algorithm can proceed with improving the payoff of the other organization, in the purpose of producing more solutions. However, in order not to introduce infinite loops, a taboo list of previously visited schedules must be kept (such schedules cannot be revisited).

The output of the algorithm is a list of grid schedules. The first (SPT) and the second schedule produced are definitely equitable. However, the equitableness of the rest of the schedules is not guaranteed. Therefore, after the above algorithm stops, the list of solutions is cleaned. All the solutions which are not equitably-optimal with respect to other solutions on the list are removed. Moreover, the algorithm may not produce all possible equitable solutions (the equitable part of the Pareto-front), because it may be trapped in a local optimum.

4.1.2 Telecommunication Systems and Computer Networks

A fair way of distribution of the bandwidth (or other network resources) among competing demands becomes a key issue in computer networks [35] and telecommunication network design in general [75, 159]. This section deals with problems of bandwidth allocation within telecommunication and computer networks (also called network dimensioning), basing on previous research described in [129]. We focus on the approaches that attempt to

provide equitable treatment of all the network flows (demands) while allocating resources [102, 124].

The problem of network dimensioning is a typical example of a fair distribution problem with a budget. Each network demand is associated with a cost that is a function of the network topology and of the location of demand endpoints in that topology. The total sum of costs for realizing a demand cannot exceed the budget. This implies that the solution of the network dimensioning problem described in this chapter can also be applied to other fair distribution problems with a budget.

Expanding demand on Internet services has led to an increased role of the traffic carried by the IP protocol in telecommunication networks. The TCP protocol is the most frequently used transport protocol in best-effort IP networks. The data traffic carried by the TCP protocol adapts its throughput to the amount of available bandwidth. Such traffic, called *elastic traffic*, is capable to use the entire available bandwidth, but it is also able to reduce its throughput in the presence of contending traffic. It should be noted here that elastic traffic communicated by the TCP protocol is currently the most significant portion of traffic in IP networks. Applications such as World Wide Web, e-mail, or Peer-to-Peer file-sharing all use the TCP protocol and therefore communicate elastic traffic, which forms the majority of the traffic volume in IP networks. Nowadays, network management often faces the problem of designing networks that carry elastic traffic. These network design problems are essentially network dimensioning problems as they can be reduced to a decision about link capacities. Flow sizes are outcomes of the design problem, since the flows adapt to given network resources on a chosen path.

Network management must stay within a budget constraint on link bandwidth to expand network capacities. An obvious goal is to achieve a high throughput of the IP network to increase the multiplexing gains (due to the use of packet switching by the IP protocol). This traffic is offered only a best-effort service, and therefore network management is not concerned with offering guaranteed levels of bandwidth to the traffic. A straightforward network dimensioning with elastic traffic could be thought of as a search for such network flows that will maximize the aggregate network throughput while staying within a budget constraint for the costs of link bandwidth. However, maximizing aggregate throughput can result in extremely unfair solutions allowing even for starvation of flows for certain demands. At the other extreme, while looking at the problem from the perspective of a network user, the network flows between different nodes should be treated as fairly as possible [16]. The so-called Max-Min Fairness (MMF) [13, 63] is widely considered as such ideal fairness criteria. Indeed, the lexicographic max-min optimization used in the MMF approach generalizes equal sharing at a single link bandwidth to any network while maintaining the Pareto optimality. Certainly, allocating the bandwidth to optimize the worst performance may cause a large worsening of the overall throughput of the network. Therefore, network

management must consider two conflicting goals: increasing throughput and providing fairness.

Any solution to the network dimensioning problem may be implemented by network management today with the use of traffic engineering techniques, for example in an IP/MPLS network. This means that it is not necessary to change the physical speeds of links, or to change any hardware configuration in the network. A new solution to the network dimensioning problem can be implemented using soft reconfiguration of the network.

The search for compromise solutions that do not starve network flows, and give satisfying levels of throughput has led to the development of methods depending on maximization of the sum of the flows evaluated with some (concave) utility function. In particular, the so-called Proportional Fairness (PF) approach [75] maximizes the sum of logarithms of the flows. The approach has been further extended to a parametric class of concave utility functions [114]. However, every such approach requires to building (or to guessing) a utility function prior to the analysis and later it gives only one possible compromise solution. More general parametric approach may depend on the use of the so-called Ordered Weighted Averaging (OWA) (see chapter 2.3.9) with weights assigned to the ordered outcomes (flows) thus allowing to model various fair preferences [125].

In this section, we shall demonstrate how the computational methods of equitable optimization can be used in practice. The problem of allocating network throughput to satisfy the demands of various services is a typical distribution problem. However, the setting of this problem is not typical for an ODS. While the resulting distribution applies to an ODS which is a telecommunication or computer network under a single administration (for example, an autonomous system), the problem is solved by a decision maker that has complete information and is able to enforce that the equitable solution chosen by him is accepted by all agents in the ODS. This decision maker is the operator of the network. This situation is not typical for ODS, where complete information and control is usually not available to a single agent. In the next sections, we shall consider other practical applications of the theory of equitable optimality that do not require complete control or information.

The Bandwidth Allocation Problem

The efficient optimization problem considered in this section is a problem of distributing network resources to demands while satisfying a general budget constraint. Each demand is associated with a cost that is a consequence of the network topology and of the location of the demand endpoints in the topology. Solving the efficient optimization problem without concern for fairness would therefore allocate network resources to the cheapest demands, as the assumption of elastic traffic ensures that these demands would use all available resources. However, some (or indeed a majority) of more expensive demands would be unsatisfied (starved).

The basic problem of network dimensioning with elastic traffic can be formulated as a Linear Programming (LP) based resource allocation model as follows [133]. Given a network topology $G =< V, E >$, consider a set of pairs of nodes as set $I = \{1, 2, \ldots, n\}$ of demands representing the elastic flow from source v_i^s to destination v_i^d. For each demand, we have a given set P_i of possible routing paths in the network from the source to the destination. This information can be summarized with binary coefficients δ_{eip}, where $\delta_{eip} = 1$, if link e belongs to routing path $p \in P_i$ (connecting v_i^s with v_i^d), and $\delta_{eip} = 0$ otherwise.

For each demand $i \in I$, the elastic flow from source v_i^s to destination v_i^d is a variable representing the model outcome and it will be denoted by x_i. This flow may be realized along various paths $p \in P_i$. The flow may be either split among several paths or a single path must be finally selected to serve the entire flow. Actually, the latter case of nonbifurcated flows is more commonly required. Both bifurcated or nonbifurcated flows may be modeled as $x_i = \sum_{p \in P_i} x_{ip}$ where x_{ip} (for $p \in P_i$) are nonnegative variables representing the elastic flow from source v_i^s to destination v_i^d along routing path p. The single-path model requires additional multiple choice constraints to enforce nonbifurcated flows. This can be implemented with additional binary (flow assignment) variables u_{ip} equal 1 if path $p \in P_i$ is assigned to serve flow x_i, and 0 otherwise. Assuming existence of some constant M upper bounding the largest possible total flow x_i, the assignment variables u_{ip} can easily be used to limit the number of positive flows x_{ip} with the following constraints:

$$0 \leq x_{ip} \leq M u_{ip}, \quad u_{ip} \in \{0, 1\} \; \forall \; i \in I; \; p \in P_i, \tag{4.1}$$

$$\sum_{p \in P_i} u_{ip} = 1 \; \forall \; i \in I, \tag{4.2}$$

The network dimensioning problem depends on allocating the bandwidth to several links in order to maximize flows of all the demands. Typically, the network is already operated and some bandwidth is already allocated (installed) while decisions are rather related to the network expansion. Therefore, we assume that each link $e \in E$ has already capacity a_e while decision variables ξ_e represent the bandwidth newly allocated to link $e \in E$, thus expanding the link capacity to $a_e + \xi_e$. Certainly, all the decision variables must be nonnegative: $\xi_e \geq 0$ for all $e \in E$ and there are usually some bounds (upper limits) on possible expansion of the links capacities: $\xi_e \leq \bar{a}_e$ for all $e \in E$. Finally, the following constraints must be fulfilled:

$$\sum_{i \in I} \sum_{p \in P_i} \delta_{eip} x_{ip} \leq a_e + \xi_e \qquad \forall e \in E, \tag{4.3}$$

$$0 \leq \xi_e \leq \bar{a}_e \qquad \forall e \in E, \tag{4.4}$$

$$\sum_{p \in P_i} x_{ip} = x_i \qquad \forall i \in I, \tag{4.5}$$

where equations (4.5) define the total demand flows, while inequalities (4.3) establish the relation between the demand flows and the link bandwidths. The quantity $y_e = \sum_{i \in I} \sum_{p \in P_i} \delta_{eip} x_{ip}$ is the load of link e and it cannot exceed the available link capacity.

Further, for each link $e \in E$, the cost of allocated bandwidth is defined. In the basic model of network dimensioning it is assumed that any real amount of bandwidth may be installed and marginal costs c_e of link bandwidth is given. Hence, the corresponding link dimensioning function expressing amount of capacity (bandwidth) necessary to meet a required link load [133] is a linear function. While allocating the bandwidth to several links in the network dimensioning process, the decisions must keep the cost within available budget B for all link bandwidths. Hence the following constraint must be satisfied:

$$\sum_{e \in E} c_e \xi_e \leq B \,. \tag{4.6}$$

The model constraints (4.3)–(4.6) together with respective nonnegativity requirements define a linear programming (LP) feasible set. It turns into Mixed Integer LP (MILP), however, if nonbifurcated flows are enforced with discrete constraints (4.1)–(4.2).

Link modularity (bandwidth granulation) is is a common feature in communications networks [133]. Therefore, in more realistic models for each link $e \in E$ the minimum unit of bandwidth b_e is assumed to be available for allocation (installation) and c_e represents the corresponding unit cost. The corresponding link dimensioning function is then a stepwise function. In the case of modular links (discrete bandwidth units b_e), the installed capacity ξ_e must satisfy an additional equation:

$$\xi_e = b_e z_e \quad \forall e \in E, \tag{4.7}$$

where z_e is an integer decision variable representing the number of bandwidth units installed at link e. The model constraints (4.3)–(4.6) extended with (4.7) turns then into MILP feasible set even if bifurcated flows are allowed. The network dimensioning model can be considered with various objective functions, depending on the chosen goal. One may consider two extreme approaches. The first extreme is maximization of the total throughput (the sum of flows) $\sum_{i \in I} x_i$. At the other extreme, network flows between different nodes should be treated as fairly as possible which leads to maximization of the smallest flow or rather to the lexicographically expanded max-min optimization (the so-called max-min ordering) allowing also to maximize the second smallest flows provided that the smallest remain optimal, the third smallest, etc.

This approach is widely recognized in networking as the so-called Max-Min Fairness (MMF) [13, 63] and it is consistent with the Rawlsian theory of justice [137]. Note that for convex models there exists at least one blocked outcome which is constant on the entire set of optimal solutions to the

Max-Min problem because of an active link capacity constraint. Hence, the MMF solution can be found by solving a sequence of properly defined Max-Min problems with fixed outcomes (flows) that have been blocked by some critical constraints (link capacities) [79, 106]. Unfortunately, in our network dimensioning model it applies only to the basic LP constraints (4.3)–(4.6). In the case of a nonconvex feasible set such a blocked quantity may not exist [123], which makes the approach not applicable to the case of nonbifurcated flows enforced by discrete constraints (4.1)–(4.2).

In the simplified problem with linear link dimensioning function (no modularities) and dimensioning of a completely new network ($a_e = 0$ for all links), the cost of the entire path p for demand i can be directly expressed by the formula $\kappa_{ip} = \sum_{e \in E} c_e \delta_{eip}$. Hence, the cheapest path for each demand can easily be identified and preselected. Constraints (4.6) and (4.3) may then be treated as equations and they allow one to eliminate variables ξ_e, thus formulating the problem as a simplified resource allocation model with only one constraint $\sum_{i=1}^{n} \kappa_i x_i = B$ and variables x_i representing directly the decisions. In the problem under consideration the cost of available link capacity is actually nonlinear (piecewise linear) and this results in the lack of direct formula for the path cost since it depends on possible sharing with other paths of the preinstalled bandwidth (free capacity a_e). Such a simplification is certainly also impossible for the modular case, due to additional discrete constraints (4.7).

In the simplified dimensioning model (with preselected paths and continuous bandwidth) the throughput maximization approach apparently would choose one variable x_{i^o} which has the smallest marginal cost $\kappa_{i^o} = \min_{i \in I} \kappa_i$ and make that flow maximal within the budget limit ($x_{i^o} = B/\kappa_{i^o}$), while eliminating all other flows (lowering them to zero). On the other hand, the MMF concept applied to the simplified dimensioning model would lead us to a solution with equal values for all the flows: $x_i = B/\sum_{i \in I} \kappa_i$ for $i \in I$. Such allocating of resources to optimize the worst performance may cause a large worsening of the overall (mean) performance as the MMF throughput ($nB/\sum_{i \in I} \kappa_i$) might be considerably smaller than the maximal throughput ($B/\min_{i \in I} \kappa_i$). In more realistic dimensioning models assuming bandwidth modularity or other nonlinearities in link dimensioning function (like the existence of a free capacity a_e of preinstalled bandwidth) and nonbifurcation requirements, a direct formula for the path cost is not available and the corresponding solutions are not so clear. Nevertheless, the main weaknesses of the above solutions remain valid. The throughput maximization can always result in extremely unfair solutions allowing even for starvation of certain flows while the MMF solution may cause a large worsening of the network throughput. In an example built on the backbone network of a Polish ISP, it turned out that the throughput in a perfectly fair solution could be less than 50% of the maximal throughput [125].

Network management may be interested in seeking a compromise between the two extreme approaches discussed above. One of the possible solutions

depends on maximization of the sum of flows evaluated with some (concave) utility function $U(\mathbf{x}) = \sum_{i \in I} u(x_i)$. A parametric class of utility functions [114]:

$$u(x_i, \alpha) = \begin{cases} x_i^{1-\alpha}/(1-\alpha) & \text{if } \alpha \neq 1 \\ \log(x_i) & \text{if } \alpha = 1, \end{cases} \tag{4.8}$$

may be used for this purpose generating various solution concepts for $\alpha \geq 0$. In particular, for $\alpha = 0$ one gets the throughput maximization which is the only linear criterion within the entire class. For $\alpha = 1$, it represents the Proportional Fairness (PF) approach [75] that maximizes the sum of logarithms of the flows while it converges to the MMF with α tending to the infinity. However, every such approach requires building (or guessing) a utility function prior to the analysis and later it gives only one possible compromise solution. It is very difficult to identify and formalize the preferences at the beginning of the decision process. Moreover, apart from the trivial case of throughput maximization, all the utility functions that really take into account any fairness preferences are nonlinear. Nonlinear objective functions applied to the MILP models we consider result in computationally hard optimization problems. In the following, we shall describe an approach that allows us to search for such compromise solutions with multiple linear criteria rather than nonlinear objective functions.

The bandwidth allocation problem we consider may be viewed as a special case of a general resource allocation problem where set I of m demands is considered and for each demand $i \in I$ its measure of realization x_i is a function $x_i = f_i(\xi)$ of allocation pattern $\xi \in A$. This function, called the individual objective function, represents the outcome (effect) of the allocation pattern for demand i. In applications we consider f_i expresses the demand flow and a larger value of the outcome means a better effect (higher service quality or client satisfaction). This leads us to a vector maximization problem:

$$\max \{(x_1, x_2, \ldots, x_m) \ : \quad \mathbf{x} \in Q\}, \tag{4.9}$$

where $Q = \{(x_1, \ldots, x_m) : x_i = f_i(\xi) \quad \text{for } i \in I,\ \xi \in A\}$ denotes the attainable set for outcome vectors $\mathbf{x}$. For the network dimensioning problems we consider the set Q is an MILP feasible set defined by basic constraints (4.1)–(4.6) with additional discrete constraints (4.7) in the case of modular bandwidth.

The Equitable Bandwidth Allocation Problem

The theory of equitable optimality and the computational methods described in section 2.3.9 can be applied to find equitable solutions to problem 4.9. In the terminology introduced in section 2.3.2, problem 4.9 is the efficient optimization problem. We shall now demonstrate that it is indeed possible to find equitable solutions for even such a complex efficient optimization problem. A similar approach can be used for other fair distribution problems with a budget. First, we shall reformulate the efficient optimization problem as a

new multicriteria problem. The solutions of the new (equitable) multicriteria problem will be equitably optimal solutions of the efficient optimization problem. Next, we shall use a method for solving the new multicriteria problem. In this section, we describe the usage of the well-known reference point method for this purpose.

Following the approach suggested in section 2.3.9, the equitable optimization problem may be formulated as a following MILP problem:

$$\max\,(\eta_1, \eta_2, \ldots, \eta_m) \qquad \text{subject to} \quad \mathbf{x} \in Q, \tag{4.10}$$

$$\eta_k = kt_k - \sum_{i=1}^{n} d_{ik} \quad \text{for } k \in I, \tag{4.11}$$

$$t_k - d_{ik} \le x_i, \quad d_{ik} \ge 0 \quad \text{for } i, k \in I\,. \tag{4.12}$$

Note that problem (4.10)–(4.12) adds only linear constraints to the original attainable set Q. Hence, for the basic network dimensioning problems with the set Q defined by constraints (4.1)–(4.6), the resulting formulation (4.10)–(4.12) remains in the class of (multi-criteria) MILP. The same applies to the modular dimensioning model with additional constraints (4.7).

Although defined with simple linear constraints, the expanded model (4.10)–(4.12) introduces m^2 additional variables and inequalities. This may cause a serious computational burden for real-life network dimensioning problems. Note that the number of demands corresponds to the number of ordered pairs of network nodes which is already on the order of the square of the number of nodes $|V|$. Thus, finally the expanded multi-criteria model introduces $|V|^4$ variables and constraints which means polynomial but fast growth and may not be acceptable for larger networks. In order to reduce the problem size one may attempt the restrict the number of criteria in the equitable optimization problem.

Let us consider a sequence of indices $K = \{k_1, k_2, \ldots, k_q\}$, where $1 = k_1 < k_2 < \ldots < k_{q-1} < k_q = m$, and the corresponding restricted form of the multi-criteria efficient optimization problem:

$$\max\,\{(\eta_{k_1}, \eta_{k_2}, \ldots, \eta_{k_q}) :\, \eta_k = \theta_k(\mathbf{x}) \quad \text{for } k \in K, \quad \mathbf{x} \in Q\}\,, \tag{4.13}$$

with only $q < m$ criteria. Following Theorem 1, the equitable optimization problem allows us to generate any fairly efficient solution of problem (4.9). Reducing the number of criteria we restrict these capabilities. Nevertheless, one may still generate reasonable compromise solutions.

Theorem 3. *If $\mathbf{x}^o$ is an efficient solution of the restricted problem (4.13), then it is an efficient (Pareto-optimal) solution of the multi-criteria problem (4.9) and it can be fairly dominated only by another efficient solution $\mathbf{x}'$ of (4.13) with exactly the same values of criteria: $\theta_k(\mathbf{x}') = \theta_k(\mathbf{x}^o)$ for all $k \in K$.*

For the proof, see [129].

It follows from Theorem 3 that while restricting the number of criteria in the equitable optimization problem we can essentially still expect a reasonably fair efficient solution and only *unfairness* may be related to the distribution of flows within classes of skipped criteria. In other words we have guaranteed some rough fairness while it can possibly be improved by redistribution of flows within the intervals $(\theta_{k_j}(\mathbf{x}), \theta_{k_{j+1}}(\mathbf{x})]$ for $j = 1, 2, \ldots, q-1$. Since the fairness preferences are usually very sensitive for the smallest flows, one may introduce a grid of criteria $1 = k_1 < k_2 < \ldots < k_{q-1} < k_q = m$ which is dense for smaller indices while sparser for larger indices and expect solution offering some reasonable compromise between fairness and throughput maximization. In our computational analysis of the network with 132 elastic flows (Section 5) we have preselected 24 criteria including the smallest 12 flows. Note that any restricted model contains criteria $\theta_1(\mathbf{x}) = \min_{i \in I} x_i$ and $\theta_m(\mathbf{x}) = \sum_{i \in I} x_i$ among others. Therefore, it is more detailed than any bicriteria combination of max-min and throughput maximization.

Example Solutions of Equitable Bandwidth Allocation Problems

The reference distribution approach described in section 2.3.9 has been tested on a sample network dimensioning problem with elastic traffic. Recall that in the case of elastic traffic, the network dimensioning procedure results in the capacities of links in a given network, and that the flows will adapt to the bandwidth available on links in the designed network. The input to a network dimensioning problem with elastic traffic consists of a network topology, of pairs of nodes that specify sources and destinations of flows, of sets of network paths that could be used for each flow, and of optional constraints on the capacities of links or on flow sizes. The user must also specify a budget for purchasing link capacity (B in (4.6)), prices of a unit of link capacity (possibly different for each link, c_e in (4.6)), and may specify module sizes and prices for a link. The given network topology may contain information about preinstalled link capacities (a_e in (4.3)): the budget is then spent on additional link capacities that extend the present capacity of links.

The network topology of the presented problem (Fig. 4.2) is patterned after the backbone network of a Polish ISP [125]. The network consists of 12 nodes and 18 links. All links have unit costs equal to one, and the budget for link bandwidth is $B = 1000$. Flows between any pair of different nodes were considered (i.e., $144 - 12 = 132$ flows). Since all links have equal costs of one, the path cost is equal to the path length (1, 2, 3 or 4 for the shortest paths in the example topology). For each flow, two alternative paths (the shortest and the second shortest) have been specified that could be used for transport. The entire flow had to travel along one of the paths with no splitting allowed (nonbifurcation formulation (4.1)–(4.2)). All flows are unbounded. However, it is clear that due to the budget constraint no flow can exceed B.

In [125] a simplified LP model has been studied without additional constraints on link capacity, with a limitation that flows could only use the shortest path, and with equal link costs, since in such a case it was simple

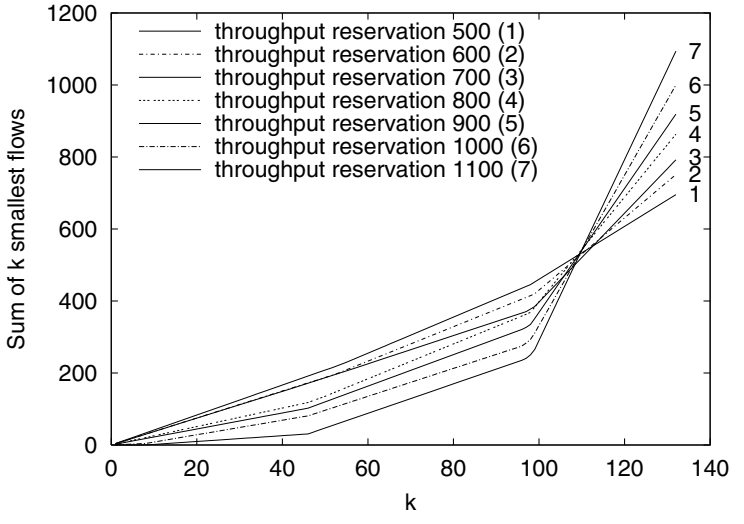


Fig. 4.2 Sample network topology patterned after the backbone network of Polish ISP

to understand the best choices with respect to fairness and overall throughput. However, for such a problem it is also simple to calculate the solution obtained by the two other methods used in literature for allocation problems with fairness objectives: max-min fairness and proportional fairness. Indeed, in [125] we have calculated these solutions and have shown that appropriate OWA aggregations allow us to obtain similar results. Additionally, using the OWA criterion, it was possible to obtain a spectrum of alternative solutions and to control the results using intuitive parameters. Here, we focus on two extensions of the problem studied in [125] that are too complex for a simple application of proportional fairness or max-min fairness. To apply either of these methods to the discussed problem extensions, it would be necessary to solve a nonlinear optimization problem or a sequence of MILP problems with changing constraints. The proposed problem modifications also make the studied models more practical and realistic.

The first studied extension allowed flows to choose one of two paths for transport (4.1)–(4.2), added constraints that limited the capacity of certain links from above and added free link capacity for certain links (4.3). The intention behind the modification was to model a situation where the network operator wishes to extend the capacity of an existing network. In this network, certain links cannot be upgraded beyond certain values due to prohibitive costs or administrative reasons (for instance, it may be cheap to use existing fiber that has not been in use before, but it may be prohibitively expensive to install additional fiber). The existence of free link capacity and of link capacity constraints may be the reason for choosing alternative paths for certain flows.

Additionally, a modular version of the original problem has been considered. In the second problem modification, flows were still limited to shortest paths, and no constraints on link capacities were added. The size of a link capacity module was set to 5. For each link, integer variable z_e has been (see (4.7)). Modular link capacities are frequently encountered in networks, when it is simple to upgrade a link by installing an equipment module that is capable of faster communication over the same link. Modular link capacities also occur in telecommunication networks that use traffic trunks, or portions of link capacity that are indivisible and therefore allocated in a modular way.

The reference point method has been chosen to solve the formulated multicriteria optimization problem. As in any application of this method, it is necessary to choose aspiration and reservation levels for each criterion. This task becomes challenging for a large number of criteria, as in this case. (However, it is still a simpler job than choosing weights for the same number of criteria, because that requires a pairwise comparison of criteria.) We shall now demonstrate how the reference point method can be applied to a complex multicriteria optimization problem.

For all model versions, the final input to the model consisted of the reservation and aspiration levels for the sums of ordered criteria. For simplicity, all aspiration levels were set close to the optimum values of the criteria, and only reservation levels were used to control the outcome flows. One of the most significant parameters was the reservation level for the sum of all criteria (the network throughput). This value denoted by η^r_m was selected (varying) separately from the other reservation levels. All the other reservation levels were formed following the linearly increasing sequence of the ordered values with slope (step) r and where the reservation level for minimal flow was $\phi_1 = 1$. Hence, for the final criteria $\eta_k = \theta_k(\mathbf{x})$ representing sums of the ordered outcomes in model (4.10)–(4.12), the sequence of reservation levels increased quadratically (except for the last one). Thus, the three parameters have been used to define the reference distribution but we have managed to identify various fair and efficient allocation patterns by varying only two parameters: reservation level η^r_m for the total throughput and slope r for the linearly increasing sequence.

While dealing with a simplified model in [125] we have used all criteria η_k which resulted in the linear program containing a large number of constraints (132^2). Here, we have limited the number of criteria η_k to 24, by choosing only the indices $1, 2, 3, \ldots, 9, 10,\ 11, 12, 18, 24, 30, 36, 48, 60, 72, \ldots, 120, 132$ from the full set of all indices. As a result, the computation time drops from around one hour for each problem to the order of seconds. At the same time, the ability to control the outcomes using the reservation levels has not deteriorated; we were able to obtain similar results with the reduced set of criteria as with the full set.

In the first experiment, we used the first model extension that introduced alternative paths for flows, free link capacity and upper limits on capacity for certain links. For certain links, free link capacity was set to values from

5 to 20, and the upper limit on the capacity of certain links was set to 20. Due to the presence of free link capacity and upper limits on link capacity, the MILP solver found solutions where certain flows had to use alternative paths rather than shortest paths. These flows were more expensive than other flows that were allowed to use their shortest paths. Recall that we have used a single-path formulation, meaning that the entire flow had to be switched to the alternative path. Flows could not be split, which is consistent with several traffic engineering technologies used today.

In the experiment, the reservation level η_m^r and the slope r were used to search for compromise solutions that traded off fairness against efficiency. The throughput reservation was varied from 500 to 1100. As η_m^r increases, the cheaper flows receive more throughput at the expense of more expensive (longer) flows. For values of η_m^r above 1100, some flows were starved, and therefore these outcomes were not considered further.

The linear increase of the other reservation levels was varied as well. The parameter r could have values of: 0.02, 0.03 and 0.04. The results of the experiment are shown in Fig. 4.3 with the corresponding generalized Lorenz curves [128]. The figures present plots of cumulated ordered flows $\theta_k(\mathbf{x})$ versus number k (rank of a flow in ordering according to flow throughput) which means that the normalizing factor $1/m = 1/132$ has been ignored (for both axes). The total network throughput is represented in the figures by the height of the right end of the curve ($\theta_{132}(\mathbf{x})$). A perfectly equal distributions of flows would be graphically depicted by an ascending line of constant slope. All other (unequal) distributions of flows are represented by convex lines. First of all, one may notice that all the lines intersect each other which guarantees that no solution fairly dominates any other solution (Theorem 1). This confirms that our approach enables us to generate various fair (fairly efficient) solutions. Due to the limited resources, any increase of the throughput reservation enforces the increase of the cheaper flows (implemented on shorter paths) while restricting the most expensive flows (longer paths) sharing the same link capacities. This appears with reversal order of the solution lines at their ends. Actually, it turns out that all the lines intersect each other around the same point of $k = 110$. Hence, the available budget essentially limits the maximum throughput of about 80% of the smallest flows to the level of about 500 which cannot be exceeded without starvation of some flows. Nevertheless, it is still possible to increase the total throughput (of all 132 flows) while decreasing the fairness (increasing differences among flows).

Note that under moderate throughput requirements, as r increases, the medium flows gain at the expense of the larger ones, thus enforcing more equal distribution of flows (one may observe flattening of the curves). On the other hand, with higher throughput reservations the larger flows are protected by this requirement and increase of r causes that the medium flows gain at the expense of the smallest flows (one may observe convexification of the curves). For values of r higher that 0.04, the increase of the throughput reservation resulted in flow starvation.

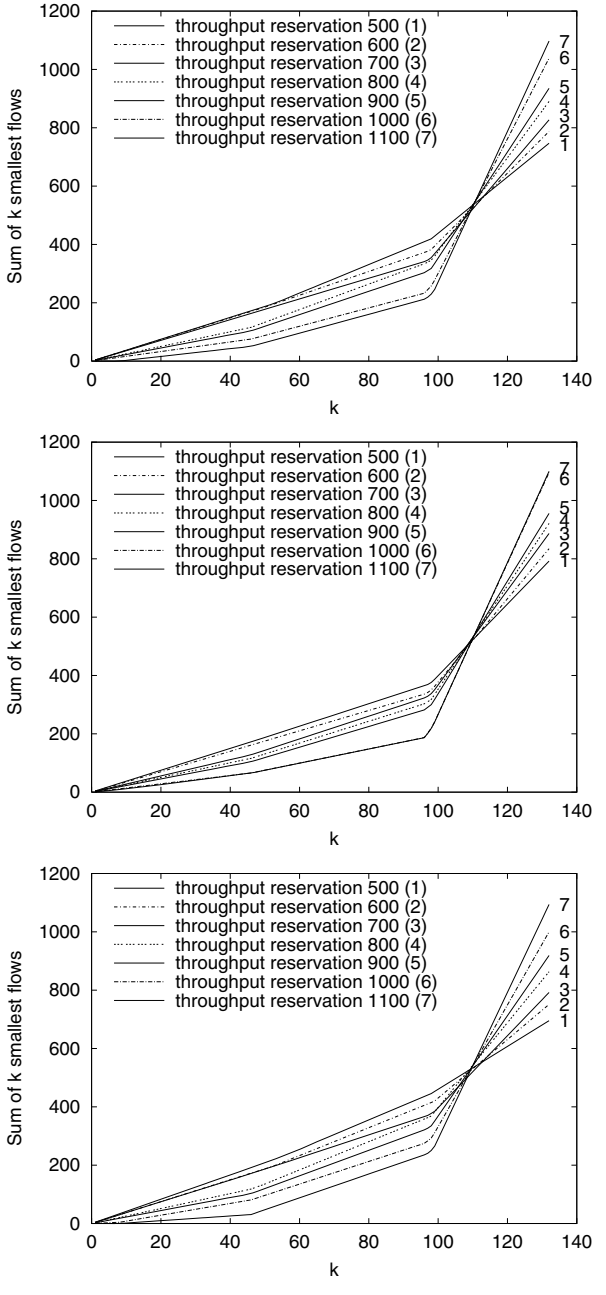


Fig. 4.3 Flow distribution for varying throughput reservation with $r = 0.02$ (a), $r = 0.03$ (b), $r = 0.04$ (c)

Observe from Fig. 4.3(c) that for $\eta_m^r = 1100$ (and for some other values of η_m^r), the boundary between the largest flows (part of the Generalized Lorenz curve with the highest slope) and the second-largest flows is not sharp. The change of slope is gradual, resembling a round knee. The reason for this is the presence of three flows that should receive the same amount of throughput as the largest flows since they are all transported on paths of the length of 1, but cannot due to the presence of upper constraints on link capacities. These flows receive as much as they can, but some capacity is left for other flows that must travel on the same constrained link. Here the solution violates fairness in the attempt to get a higher total network throughput.

Also, note in the same figure that the boundary between the smallest flows for $\eta_m^r = 500$ and for $\eta_m^r = 1100$ is not in the same position. The reason for this is once again the upper constraint on link capacities. For $\eta_m^r = 500$, there are 8 flows that should be in the middle group of flows but cannot, since flows in the middle group receive so much throughput that the constraints on link capacity would be violated. Therefore, these flows are downgraded to the group of smallest flows and receive the same amount of throughput as the smallest flows – here the solution preserves fairness.

Note that the throughput reservation was effectively used to find outcomes with the desired network throughput. On the other hand, especially for large throughput reservations, the optimization procedure automatically found outcomes that divided flows into four categories according to their path costs. This shows that the presented methodology is cost-aware, and that it is possible to guarantee fairness to all flows with the same path cost (if link capacity constraints do not interfere). For the lowest throughput reservation of $\eta_m^r = 500$ and $r = 0.04$, the outcome was close to a perfectly fair distribution. Using the methodology described in this section, the user can choose from a large number of different outcomes and control the trade-off between fairness and efficiency. For the second experiment we used a slightly different dimensioning problem specification. Namely, we added the modular link capacities (4.7) into the model while eliminating the routing decisions by restriction of all demand flows to the corresponding shortest paths. Thus, the model was still in the class of MILP but with a different discrete structure. We have repeated a search for compromise solutions using similar preference parameter configurations as in the first experiment, although taking into account the constraints on modular bandwidth, the throughput reservation was varied from 450 to 800. The resulting flow distributions for the reservation slope parameter $r = 0.02$ are presented in Fig. 4.4. Predictably, the introduction of modular link capacities makes it more difficult to find fair solutions. The outcome for $\eta_m^r = 450$ is close to a perfectly even (fair) distribution, although the right end of the curve turns slightly upward. This indicates that the excess capacities of modules were used by the cheapest flows, leading to a higher network throughput than in the case of a problem without modular link capacities. On the other hand, the flows with the cheapest paths were not equal for some outcomes.

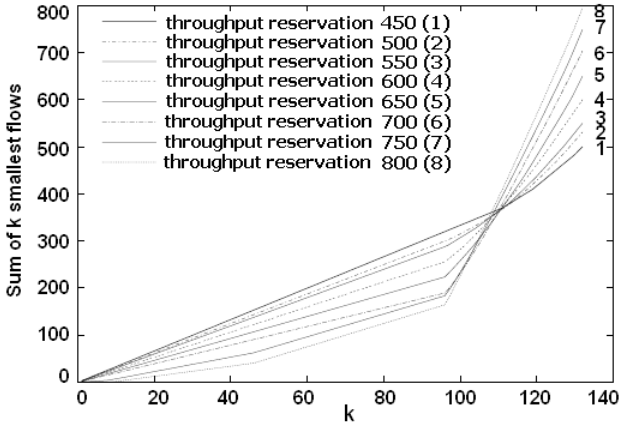


Fig. 4.4 Varying throughput reservation with $r = 0.02$ for the model with bandwidth modularity

Overall, the experiments on the sample network topology demonstrated the versatility of the methodology described. The use of reservation levels, controlled by a small number of simple parameters, allowed us to search for solutions best fitted to various possible preferences of a network designer. Using an appropriate reference point procedure, one should be able to easily find a satisfactory fair and efficient allocation pattern in a few interactive steps. The same method of finding equitable solutions to a complex efficient optimization problem can be reused in other domains.

4.1.3 Peer-to-Peer Computing

The problem of fairness management in P2P and grid systems concerns the fairness of access and provision of shared resources. In P2P file sharing, these resources are the access bandwidth of peers. In grids, resources can be CPU time or more generally, processing time of grid tasks. Unfair behavior in P2P and grid systems is often called *free-riding*. Peers or grid nodes can sometimes use resources of others without providing resources in return. The goal of fairness management is the increase of fairness in the distribution of used and provided resources. The two distributions are often combined using a distribution of the ratio of used and provided resources. In P2P systems, fairness management works by providing incentives for peers to provide resources to the system. The decentralization of control in P2P systems makes it difficult to manage fairness. Peers can only attempt to control the behavior of others using individual strategies, like in a non-cooperative game. P2P fairness

management considers the following measures of fairness of individual peers: sharing ratio and altruistic provision.

4.1.3.1 Fairness Management in P2P File Sharing

Sharing ratio is defined as the total number of uploaded bytes divided by total bytes downloaded [116]. A P2P Fairness Management systems' primary objective is to keep the sharing ratio above a certain level for every peer. A sharing ratio of 1 and above is considered fair, since it indicates that a peer uploads the same amount (or more) as it has downloaded. If the sharing ratio drops below a certain level it is possible to exclude the peer from the system.

Altruistic provision is defined as the difference between the expected upload rate and the download rate [116, 131]. Piatek et al. also gives an alternative definition of altruistic provision: an upload contribution that can be withdrawn without loss in download performance of the providing peer, given that the other peers use a strategy that is based on the peer's upload contribution.

To ensure a fair sharing ratio, BitTorrent uses the *choke algorithm* described in [26, 131, 89]. This algorithm is derived from the game theoretic *Tit-for-Tat* strategy and affects the peer selection process. It helps to choose the most reciprocating peers, but also prevents the low bandwidth peers from starvation (optimimistic unchoke). The algorithm attempts to prevent malicious peers from free-riding.

An example of a modified Bittorrent client is the BitTyrant project[131], which merges several greedy techniques. The BitTyrant client exploits the original BitTorrent algorithms by altering the reciprocity factor. Authors describe alternative choking algorithms. [26] evaluated the BitTyrant modifications. They discovered that BitTyrant's modified algorithms have an unexpected positive impact on system performance.

[94] proposed other modifications of the BitTorrent algorithm. Their client maximize client download rate. An extension of their work was made by [100]. They described different techniques to maximize the peer download rate. Locher et al. designed and implemented a BitTorrent client called BitThief that exploits the original choking algorithm. Their client increases its neighborhood set as much as possible to boost the chance of being unchocked by another peer.

Recently, [113] proposed the BarterCast reputation algorithm. This algorithm uses an epidemic protocol for peer discovery and download statistics exchange. The authors describe how to adapt the BarterCast algorithm to the BitTorrent protocol. Each peer sends only her own statistics (download and upload information) to known peers. Peers gather observations and reports received from other known peers. To compute the reputation of third

party peers, the MAXFLOW algorithm (Ford-Fulkerson) is used. The authors modify the BitTorrent unchoke algorithm by taking into account the reputation of a peer. Peers only assign the upload slots to peers that have a reputation above a certain threshold.

The BarterCast algorithm is also used in the TRIBLER system [135]. TRIBLER is a modification of the BitTorrent client that allows to use acquaintances ("helpers") to support a peer's download. These acquaintances are obtained through a social network.

4.1.3.2 Adversary Strategies Against P2P Fairness Management

[131] and [94] discuss some cheating strategies such as:

Exploiting optimistic unchokes: this technique uses the whitewashing attack to increase the chance of receiving an optimistic unchoke.

Downloadnig only from seeds [75]: this method is simple, because seeds do not need any reciprocation from downloading peers (since they have completed their download). On the other hand, there are usually far fewer seeds, and this method therefore reduces the parallelism of downloads.

Falsifying block availability is another exploitation of the unchoke algorithm. A peer has a greater chance of being unchoked if it offers more blocks to others [94], even if the blocks contain garbage data.

4.2 Procedural Fairness Management

4.2.1 Fairness Enforcement in P2P Multi-player Games

The question of providing procedural fairness to agents in an ODS is of course relevant in many applications. However, P2P multi-player games are a good example that adequately illustrates the complexity of the problem. In this type of game, central control over all agent actions is impossible; on the other hand, agents can gain significant advantage by acting in a way that violates the fair procedures (that are well defined as the rules of the game).

Role-playing games constitute a category of multi-player games, where the player assumes the role of a character in a virtual world. The game is most often based on some exciting stories or fables, where the players compete by playing the role of such characters as: humans, elves, magicians or priests. The idea of a typical game is to accomplish various missions or quests with the controlled character, which involves certain typical types of operations. The character travels within the virtual world and interacts with other characters or objects. The knowledge of the virtual world possessed by the player may either increase with time as the player travels or remain constrained to just the closest surroundings. Early in the game, most of the virtual world

remains unknown and the player has no knowledge of where objects are located. By collecting objects and interacting with other characters (fighting, making friends, etc.) a player may, for instance, improve certain skills, accumulate experience, earn money or collect weapons or magic spells. All these properties can be used to improve one's position over competing players. In most RPG games the player accumulates ranking points that represent the result of his past competition with other characters. The objective is to survive within the virtual world and gain a better ranking than other game players.

Massive Multiplayer Online games allow many players to interact using a client-server model. The virtual world, as well as the player's characters, is managed by a central server. In many games, there can be many servers that maintain separate virtual worlds. However, a single server's scalability is limited to tens of thousands of users (even for commercial versions of the game). MMO game servers support a considerably higher number of users than multiplayer versions of other, more interactive games (such as first-person shooters). This improved scalability is achieved by limiting the scope and type of interactivity of the game [80], and by dividing the large number of players into separate game groups (sessions) with a maintainable number of players each. This is reasonable for types of games where the virtual world can be divided in multiple parallel sub-levels between which players move for instance in a promotion-based manner (Warcraft III, Quake). Despite this limitation, MMO games are very popular and offer an attractive gaming experience. The most prominent examples of MMO RPG games include: EverQuest, Ultima Online, There.com, Star Wars Galaxies, The Sims Online or Warcraft III.

At present, scalability issues in Massive Multiplayer Online (MMO) games are usually addressed with large dedicated servers or even clusters. According to white papers of a popular multi-player online game - TeraZona [160] - a single server may support 2000 to 6000 simultaneous players, while cluster solutions used in TeraZona support up to 32 000 concurrent players. The client-server approach has a severe weakness, which is the high cost of maintaining the central processing point. Such an architecture is too expensive to support a set of concurrent players that is by an order or two orders of magnitude larger than the current amounts. To give the impression of what scalability is needed - games like Lineage report up to 180 000 concurrent players in one night.

Massive Multiplayer Online games can benefit from the application of the P2P model. However, in a P2P MMO game, issues related to procedural fairness and trust become of crucial importance, as shall be shown later in this section. How can a player be trusted not to modify his own private state to his advantage? How can a player be trusted not to look at the state of hidden objects? How can a player be trusted not to lie, when he is accessing an object that cannot be used unless a condition that depends on the player's private state is satisfied? In this section, we show how all of these questions

can be answered, and present the proposed protocols in detail. We also address performance and scalability issues of the proposed procedural fairness management mechanisms, validating the proposed architecture in a practical implementation of a P2P MMO game. This section relies on previous research [172].

The difficulties of ensuring procedural fairness in a P2P MMO game could be similar for other applications. Consider for example a Peer-to-Peer auction. In such an auction, knowing the bids of other agents would be a great advantage to a their competition, so the bids of agents would have to be kept secret. On the other hand, if these bids are stored by the peers that made them, then they can be modified when bids of the competition are revealed. This demonstrates that the auction bids are private state in a sense that is similar to the P2P MMO game, and a similar solution can be used to ensure procedural fairness for Peer-to-Peer auctions.

The proposed procedural fairness management (FM) architecture relies on cryptographic mechanisms that allow players to verify the fairness of moves. The architecture is designed for P2P MMO games, but makes use of trusted central components. It is a hybrid P2P system, the result of a compromise between the P2P and client-server models. A full distribution of the fairness management control would be too difficult and too expensive. On the other hand, a return to the trusted, centralized server would obliterate the scalability and performance gains achieved in the P2P MMO game. Therefore, the proposed compromise tries to preserve performance gains while guaranteeing procedural fairness of the game. To this end, our fairness management architecture does not require the use of expensive encryption, which could introduce a performance penalty.

A prototype of the proposed fairness management architecture [177] has been implemented in a P2P MMO game. The implementation uses FreePastry, a public-domain implementation of Pastry, as the overlay network, and Scribe for application-layer multicast. The implemented game can be played both on PCs and on mobile phones connected to a Bluetooth access point. The overlay joins both types of nodes together. The performance of implemented fairness management mechanisms has been tested using experiments with the prototype.

4.2.1.1 P2P MMO Games

Several multi-player games (MiMaze, Age of Empires) have already been implemented using the P2P model. However, the scalability of such approaches is in question, as the game state is broadcasted between all players of the game. AMaze [12] is an example of an improved P2P game design, where the game state is multicast only to nearby players. Still in both cases, only the issue of public state maintenance has been addressed. The questions of how to deal with the private and public concealed states have not been answered (see section 4).

The authors of [11] have proposed a method of private state maintenance that is similar to ours. They propose the use of commitments and of a trusted "observer", who verifies the game online or at the end of the game. However, the authors of [11] have not considered the problem of concealed or conditional state. Therefore, their procedural FM architecture is incomplete. Also, the solution proposed in [11] did not address games implemented in the P2P model. In [60], the subject of fair ordering of public state updates has been considered which makes it impossible for players to cheat, while taking into account different network communication delays. This method can be used in P2P games that use peers to manage public objects, such as in the design of [80].

The work of Knutsson on P2P support for MMO games [80] offers an interesting perspective on implementing Massive Multiplayer Online games using the P2P model. The approach presented mostly addresses performance and availability issues, while leaving many security and trust issues open. In this section, we discuss protocols that can be applied in order to considerably improve the design of Knutsson in terms of security and procedural Fairness Management. In [60], a P2P approach to MMO games has also been proposed, again without taking into account issues of security and procedural Fairness Management.

Reputation-based mechanisms could be used in P2P games. A player would receive a reputation based on a history of previous games, and this reputation could be used to exclude cheating players from a game. The reason for the use of reputation mechanisms in many networked applications is the lack of enforcement mechanisms that could be used to provide procedural FM, as noted in [118].

Any reputation system has certain systematic drawbacks which is why it may be worth avoiding a reliance on reputation systems in P2P games. Among these, the most important is the problem of first-time cheating. Any peer may build up a high reputation and then cheat in order to win in a game (for example, behave fairly in unimportant encounters, and cheat during a critical encounter).

Cryptography has considered the problems of fair agreements, games, and multi-party computations. There has been research on the problem of "mental poker", or fair implementation of a distributed poker game [169]. The protocol allows for fair drawing of cards in a poker game, however, it assumes that the game players do not leave the game and is therefore unsuitable for P2P applications. Protocols on fair agreement using a third party are utilized in our research. Multi-party computation allows us to calculate an algebraic function of inputs supplied by several parties without revealing the inputs and without centralized control. The research so far in this area requires that the function should be expressed as a Boolean circuit. Applications of multi-party computation in our research are a problem of future work.

4.2.1.2 Adversaries in P2P MMO Games

The adversary strategies described in this section illustrate some of the security and procedural FM weaknesses of P2P game implementations so far. We shall use a working assumption that the P2P MMO game uses some form of Dynamic Hash Table (DHT) routing in the overlay network, without assuming a specific protocol. In the following section, we describe a procedural FM architecture that can be used to prevent the attacks described in this section.

Private State: Self-modification

P2P game implementations that allow a player to manage their own private state [80] do not exclude the possibility that a game player can deliberately modify his own private state (e.g. experience, possessed objects, location, etc.) to gain advantage over other game players. A player may also alter decisions already made in the past during player-player interaction that may affect the outcome of such an interaction. A similar adversary strategy can be used in other applications, such as Peer-to-Peer auctions, where a peer may modify his own bid in order to win the auction.

Public State: Malicious/Illegal Modifications

In a P2P MMO game, updates of public state may be handled by a peer who is responsible for a public object. The decision to update public state depends then solely on this peer - the coordinator. Furthermore, the coordinator may perform malicious modifications and multicast illegal updates to the group. The falsified update operation may be directly issued by the coordinator and returned back to the group as a legal update of the state. Such an illegal update may also be issued by another player that is in a coalition with the coordinator, and accepted as a legal operation. Once again, in a Peer-to-Peer auction, the coordinator can maliciously change the auction outcomes by changing the public description of an object in a way that would discourage competing agents from bidding (although colluding bidders would know the real value of the object).

Attack on the Replication Mechanism

When state is replicated in a P2P game, replication players are often selected randomly (using the properties of the overlay to localize replicated data in the virtual network). This can be exploited when the replication player can directly benefit from the replica of the knowledge he/she is storing (i.e. the replication player is in the region of interest and has not yet discovered the knowledge by himself). In P2P auctions, attacks on replication mechanisms can result in knowledge of rival bids.

Attack on P2P Overlays

In a P2P overlay (such as Pastry), a message is routed to the destination node through other intermediary nodes. The messages travel in open text and can be easily eavesdropped by competing players on the route. The eavesdropped information can be especially valuable if a player is revealing his own private state to some other player (player-player interaction). In such case, the eavesdropping player will find out whether the interacting players should be avoided or attacked.

The malicious player may also deliberately drop messages that he is supposed to forward. Such an activity will obstruct the game to some extent, if the whole game group is relatively small.

Conclusion from Described Attacks

Considering all of the attacks described in this chapter, a game developer may be tempted to return to the safe model of a trusted, central server. The purpose of this article is to show that this is not completely necessary. The procedural FM architecture presented in the next section will require trusted centralized components. However, the role of these components, and therefore, the performance penalty of using them, can be minimized. Thus, the achieved architecture is a compromise between the P2P and client-server models that is secure and benefits from increased scalability due to the distribution of most game activities.

4.2.1.3 Fairness Management Architecture

The procedural FM architecture proposed in this section is visualized in Figure 4.5. It uses several cryptographic primitives such as commitment protocols and secret sharing. It also uses certain distributed computing algorithms, such as Byzantine agreement protocols. The relationships between the components of the procedural FM architecture will be described in this section. Note that the modular design of our the architecture makes it

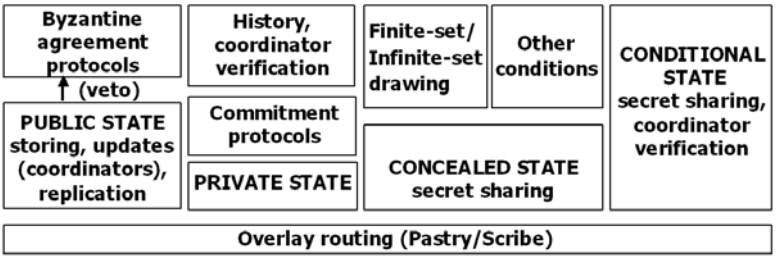


Fig. 4.5 A fairness management architecture for P2P MMO games

possible to implement selected procedural FM mechanisms; this is the approach that has been chosen in our prototype implementation.

Our procedural FM architecture for P2P MMO games will use partitioning of game players into groups, as in the approach of [80]. A group is a set of players who are in the same region. All of these players can interact with each other. However, players may join or depart from a group at any time. Each group must have a trusted coordinator, who is not a member of the group (he can be chosen among the players of another region or be provided by the game managers). The coordinator must be trusted because of the necessity of verifying private state modifications (see below). However, the purpose of the procedural FM architecture is to limit the role of the coordinator to a minimum. Thus, the performance gains from using the P2P model may still be achieved, without compromising security or decreasing trust. In an implementation of the proposed procedural FM architecture, complexity can be decreased by assigning more functions to the trusted coordinator. However, this makes the implementation more centralized.

4.2.1.4 Game Play Scenarios Using Proposed Procedural FM Architecture

Let us consider a few possible game play scenarios and describe how the proposed procedural FM mechanisms would operate. In the described game scenarios that are typical for most MMO games, the game state can be divided into four categories:

Public state is all information that is publicly available to all players and such that its modifications by any player can be revealed.

Private state is the state of a game player that cannot be revealed to other players, since this would violate the rules of the game.

Conditional state is state that is hidden from all players, but may be revealed and modified if a condition is satisfied. The condition must be public (known to all players) and cannot depend on the private state of a player.

Concealed state is like conditional state, only the condition of the state's access depends on the private state of a player.

Note that these types of state may also be used by other applications in ODS. For example, an e-commerce system may provide agents with the ability to bid for items. The agents' bids would then be conditional state.

Player Joins a Game

From the bootstrap server or from a set of peers, if threshold PKI is used, the player must receive an ID and a public key certificate C=ID, Kpub, sjoin (where sjoin is the signature of the bootstrap server (or the peers), and Kpub is the public key that forms a pair with the secret key kpriv,) that allows strong and efficient authentication (see next section. Note that the

keys will not be used for data encryption, only for signatures). The player selects a game group and reports to its coordinator (who can be found using DHT routing). The coordinator receives the player's certificate. The player's initial private state (or the state with which he joins the game after a period of inactivity) is verified by the coordinator. The player receives a verification certificate (VC) that includes a date of validity and is signed by the coordinator.

Player Verifies His Private State

In a client-server game, the game server maintains all private state of a user, which is inefficient. In the P2P solution, each player can maintain his own private state [80], causing procedural FM problems. We have tried to balance between the two extremes. It is true that a trusted entity (the coordinator) must oversee modifications of the private state. However, it may do so only infrequently. Periodically or after special events, a player must report to the coordinator for verification of his private state. The coordinator receives the initial (recently published) private state values and a sequence of modifications that he may verify and apply on the known private state. For each modification, the player must present a proof. If the verification fails, the player does not receive a confirmation of success. If it succeeds, the player is issued a VC that has an extended date of validity and is signed by the coordinator. Verification by a coordinator is done by "zeplaying" the game of the user from the time of the last verification to the present. The proofs submitted by the player must include the states of all objects and players that he has interacted with during the period.

Note that a verification may be performed for just a part of the private state, and the issued VC may specify which elements of the private state have been verified. Verification may also, for efficiency reasons, be performed for just a random part of the period, if the player submits all intermediate state changes.

After verification, the player maintains and modifies his own state. As shall be explained below, the player collects testimonies from other players that are sufficient to prove the correctness of his private state modifications.

Player Interacts with a Public Object

Peer-to-peer overlays (like DHTs) provide an effective infrastructure for the routing and storing of public knowledge within the game group. Any public object of the game is managed by some peer. The player issues modification requests to the manager M of the public object. The player also issues a commitment of his action A that can be checked by the manager. A commitment protocol works by publishing a value (the commitment) for each operation on the private state. The commitment could be, for example, a hash function of an object. The commitment is used when a player needs to prove that his

private state has not been illegally modified. Let us denote the commitment by C(ID, A) (the commitment could be a hash function of some value, signed by the player).

Commitments should be issued whenever a player wishes to access any object, and for decisions that affect his private state. Commitments may also be used for random draws [176]. It will be useful to regard commitments as modifications of public state that is maintained for each player by a peer that is selected using DHT routing, as for any public object.

The request includes the action that the player wishes to execute, and the player's validation certificate, VC. Without a valid certificate, the player should not be allowed to interact with the object. If the certificate is valid, and the player has issued a correct commitment of the action, the manager updates his state and broadcasts an update message. The manager also sends a signed testimony T=t, A, Si, Si+1, P, sM to the player. This message includes the time t and action A, state of the public object before (Si) and after the modification (Si+1) and some information P about the modifying player (e. g., his location). The player should verify the signature sM of the manager on the testimony. The manager of the object then sends an update of the object's state to the game group.

If any player (including the modifying player, if T is incorrect) rejects the update (issues a veto), the coordinator sends T to the protesting player, who may withdraw his veto. If the veto is upheld, a Byzantine agreement round is started. (This kind of Byzantine agreement is known as the Crusader's protocol.) Note that if a game player has just modified the state of a public object and has not yet sent an update, he may receive another update that is incorrect, but will not veto this update, but send another update with a higher sequence number.

To decide whether an update of the public state is correct, players should use the basic physical laws of the game. For example, the players could check whether the modifying player has been close enough to the object. Players should also know whether the action could be carried out by the modifying player (for example, if the player cuts down a tree, he must possess an axe). This decision may require knowledge of the modifying player's private state. In such a case, the modification should be accepted if the modifying player will undergo validation of his private state and present a validation certificate that has been issued after the modification took place. In other words, the modification must be checked by a trusted entity: the group coordinator. This approach, used in the prototype implementation, can be used instead of Byzantine agreements whenever a veto is issued.

Player Executes Actions That Involve Randomness

For example, the player may search for food or hunt. The player uses a fair random drawing protocol [176] (usually, to obtain a random number). This involves the participation of a minimal number (for instance, at least 3) of

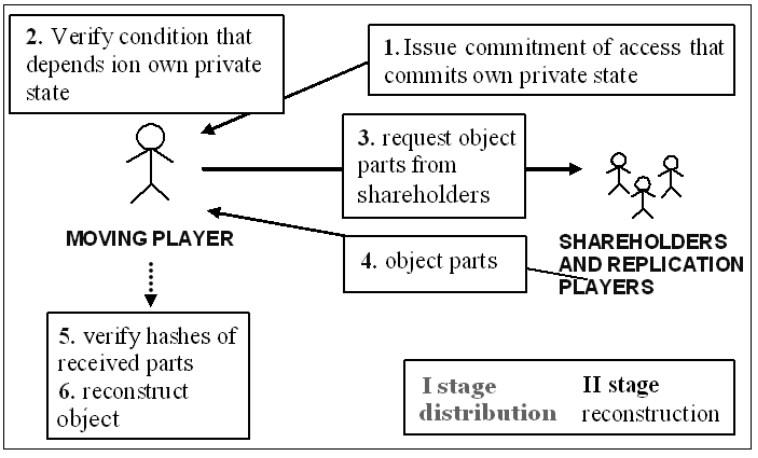


Fig. 4.6 Reconstruction of concealed and conditional state

other players that execute a secret sharing together with the drawing player. The drawing player chooses a random share l0 and issues a commitment of his share C(ID, l0) to the manager of his commitments (that are treated as public state). The drawing player receives and keeps signed shares p1,...,pn from the other players, and uses them to obtain a random number. The result of the drawing can be obtained from information that is part of constant game state (drawing tables).

Player Meets and Interacts with Another Player

For example, let two players fight. The interaction must be overseen by an arbiter, who can be any player. The two players should first check their validation certificates and refuse the interaction if the certificate of the other player is not valid. The VCs of the players are also examined by the arbiter to avoid a scenario where one of the players denies the interaction by falsely claiming that his opponent's VC is invalid. Before the interaction takes place, both players may carry out actions A1,...,Ak that modify their private state (like choosing the weapon they will use). The players must issue commitments of these actions. The commitments must also be sent to the arbiter. The actions remain secret until they can be used to improve the game result of a player. Then, the actions and relevant private state are revealed, and commitments can be used to prove correctness. The arbiter will record the commitments and the revealed actions (as shown on Figure 4.7. After the interaction is completed, the arbiter will send both players a signed testimony about the interaction.

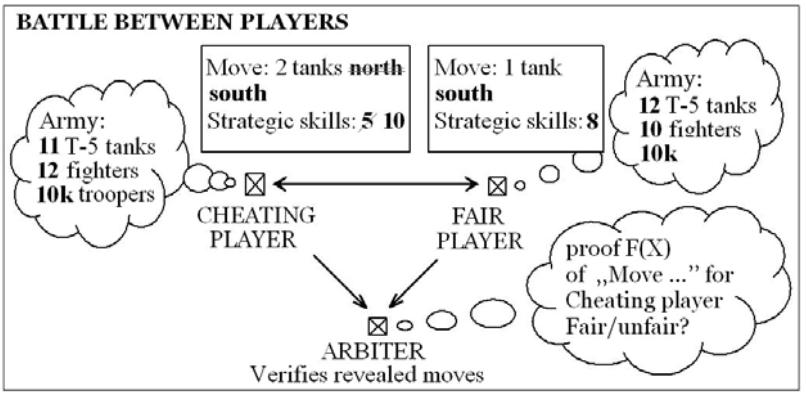


Fig. 4.7 Preventing self-modification of private state

If the interaction involves randomness, the players draw a common random number using a fair drawing protocol (they both supply and reveal shares; shares may also be contributed by other players).

Finally, the players reveal their actions to each other and to the arbiter. The results of the interaction are also obtained from fixed game information and affect the private states of both players. The players must modify their private states fairly, otherwise they will fail verification in the future (this includes the case of when a player dies. Player death is a special case. It is true that once a player is dead, he can continue to play until his VC expires. This can be corrected if the player who killed him informs the group about his death. Such a death message forces any player to undergo immediate verification if he wishes to prove that he is not dead). Note that at any time, both players are aware of the fair results of the interaction, so that a player who has won the fight may refuse further interactions with a player who decides to cheat.

Player Executes an Action That Has a Secret Outcome

For example, the player opens a chest using a key. The chest's content is conditional state - other players should not be aware of what is inside the chest. The player will modify his private state after he finds out the chest's contents. To determine the outcome, the player will reconstruct conditional or concealed state.

Conditional and concealed state can be managed using secret sharing and commitment protocols, as described in [176]. The protocol developed in [176] concerned drawing from a finite set, but can be extended to handle any public condition. However, concealed state has not been considered in our previous research on fairness of P2P games. Therefore, we needed to extend and modify our results in order to provide procedural FM of concealed state.

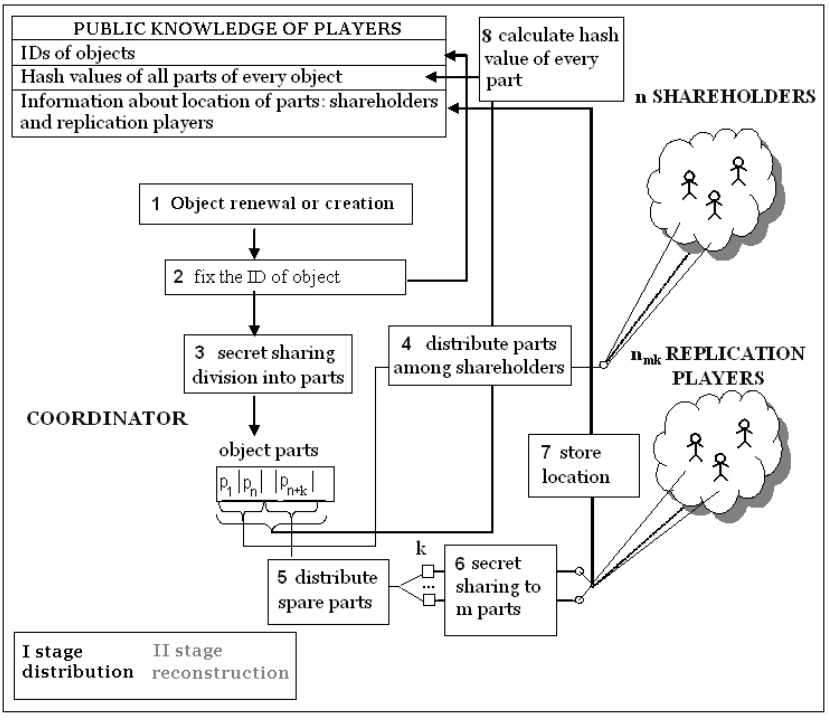


Fig. 4.8 Distribution of concealed and conditional state

The protocol for concealed and conditional state management has been shown on Figures 4.8 and 4.6. It is divided into two phases. The distribution phase needs to be executed by the coordinator whenever a concealed public object is renewed or created (step 1 on Figure 4.8). The coordinator divides the object into a fixed number of shares (step 3) that are sent to chosen players (shareholders in step 5), and to a number of replication players who are not members of the group (this is done in step 4 to decrease the likelihood of coalitions). The coordinator also calculates hash values of the shares that are public state (step 8). Apart from the initializing of the state, the coordinator does not participate in its management.

The shareholders may leave the game at any time, and the object parts from the replication players are used instead. If there are not enough parts to reconstruct an object, the object must be renewed by the coordinator. This replication approach is more resistant to coalitions than the approach proposed in [80] (relying on Pastry). Replicating the object parts by random players from the group increases the likelihood that one shareholder will receive more than one part. Then, this shareholder may form a coalition with a smaller number of other players in order to reconstruct the object. Also, the backup mechanism used in [80] may increase the number of object parts kept

by a single player. The protocol described here can also be used for random draws from a finite set (see [176]).

The second phase of the protocol is the reconstruction phase. In step 1 in Figure 4.6, the player issues a commitment of his action C(ID, A) that is checked by the shareholders. If the condition is public but depends on the player's private state, the player decides himself in step 2 whether the condition is fulfilled (he will have to prove the condition's correctness during verification in the future). In this case, the issued commitment must also commit the player's private state before the object is accessed (the player must also keep a copy of the relevant private state). If the condition to access an object is secret, the condition itself should be treated as a conditional public object. After the player checks that the condition is fulfilled, he requests and receives the object shares (steps 3 and 4) and, after verifying that the shares are correct, reconstructs the concealed or conditional object (steps 5 and 6).

When the player has reconstructed the object, he must keep the shares for verification. At the same time, the testimony issued by the shareholders will include a value of the condition that will allow the coordinator to verify the answer of the player. The elements of private state that may constitute a zero-knowledge condition should be defined during game design.

Note that concealed and conditional public objects can have states that are modified by players. If this is the case, then each state modification must be followed by the distribution phase of the protocol for object management.

4.2.1.5 Security Analysis of FM in P2P MMO Games

P2P

In this section, the attacks illustrated in previous paragraphs will be used to demonstrate how the proposed protocols protect the P2P MMO game.

Private State: Self-modification

Self-modification of private state can concern the parameters of a player, the player's secret decisions that affect other players, or results of random draws. The first type of modification is prevented by the need to undergo periodic verification of a private player's parameters. The verification is done by the coordinator on the basis of an audit trail of private state modification that must be managed by any player. Each modification requires proof signed by third parties (managers of other game objects, arbiters of player interactions). Any modification that is unaccounted for will be rejected by the coordinator. Players may verify that their partners are fair by checking a signature of the coordinator on the partner's private state. If a player tries to cheat during an interaction with another player by improving his parameters, he may succeed, but will not pass the subsequent verification and will be rejected by other players.

Modification of player's move decisions or results of random draws is prevented by the use of commitment protocols. The verification is made by an arbiter, who can be a randomly selected player. The verification is therefore subject to coalition attacks; on the other hand, making the coordinator responsible for this verification would unnecessarily increase his workload.

Note that in order for verification to succeed, the coordinator must possess the public key certificates of all players who have issued proof about the player's game. (If necessary, these certificates can be obtained from the bootstrap server). However, the players who have issued testimony need not be online during verification.

A player may try to cheat the verification mechanism by "forgetting" the interactions with objects that have adversely affected the player's state. This approach can be defeated in the following way. A player that wishes to access any object may be forced to issue a commitment in a similar manner as when a player makes a private decision. The commitment is checked by the manager of the object and must include the time and type of object. Since the commitment is made prior to receiving the object, the player cannot know that the object will harm him. The coordinator may check the commitments during the verification stage to determine whether the player has submitted information about all state changes.

Public State: Malicious/Illegal modifications

We have suggested the use of Byzantine algorithms further supported by a veto mechanism (Crusader's protocol) to protect public state against illegal/ malicious modifications. Any update request on the public state shall be multicast to the whole game group. The Byzantine verification within the group shall only take place when at least one of the players vetoes the update request of some other player. The cheating player as well as the player using the veto in unsubstantiated cases may both be penalized by the group by exclusion from the game. Such a mechanism will act largely as a preventive and deterring measure, introducing the performance penalty only on an occasional-basis.

The protection offered by Byzantine agreement algorithms will be discussed further in section 4.2.3. We believe the protection offered by such algorithms is far more secure than a coordinator-based approach and tolerable in terms of performance.

Attacks on P2P Overlays

In our security architecture, players do not reveal sensitive information. A player does not disclose his own private state, but only commitments of this state. Concealed or conditional state is not revealed until a player receives all shares. If the P2P overlay is operating correctly and authentication is used to prevent Sybil attacks, the P2P MMO game should be resistant to eavesdropping by nodes that route messages without resorting to strong encryption. A

secure channel is needed during the verification of a players private state by the coordinator.

Concealed State: Attack on Replication Mechanism

Concealed state, as well as any public state in the game, must be replicated among the peers to be protected against loss. The solution offered in [80] uses the natural properties of the Pastry network to provide replication. However, we have questioned the use of this approach for concealed state, where the replicas cannot be stored by a random peer. The existence of concealed state has not been considered in [80], and therefore the authors did not consider the fact that replicas may reveal the concealed information to unauthorized players.

In our approach for replication of concealed state, replication players are selected from outside the game group. This eliminates the benefits offered by Pastry network. On the other hand, this approach also eliminates the security risks. Please note that in our approach a certain number of players must participate to uncover specific concealed information. Therefore, a coalition with the replication player is not beneficial for a player within the game group.

Revocation of Verification Certificates

Verification is performed by trusted group coordinators that can be superpeers operated by the game provider. However, it is possible that the a malicious player could somehow set up a malicious coordinator and therefore defeat the verification mechanism. To avoid this, the superpeers should be equipped with public key certificates that are signed by other superpeers (using a Web-of-Trust model, or by a signle central authority). When a Verification Certificate is examined by a player, the player should check the superpeers signature. To do this, the player should obtain or posess the superpeers public key certificate, and validate this certificate.

4.2.1.6 Performance Analysis of FM in P2P MMO Games

Analysis of the Fairness Management Architecture

We have tried to manage trust in a P2P MMO game without incurring a performance penalty that would question the use of the P2P model. However, some performance costs are associated with the proposed mechanisms. Our initial assumption about partitioning of game players into groups (sets of players who are in the same region) is required for good performance.

Byzantine protocols have a quadratic communication cost, when a player disagrees with the proposed decision. Therefore, their use in large game groups may be prohibitive. This problem may be solved by restricting the Byzantine agreement to a group of superpeers that maintain the public state (an

approach already chosen by a few P2P applications, such as OceanStore). Another possibility is the use of hierarchical Byzantine protocols that allow the reduction of cost but require hierarchy maintenance.

Since private state is still managed by a player, it incurs no additional cost over the method of Knutsson. The additional cost is related to the verification of a player's private state by a coordinator. The coordinator must "replay" the game of a player, using provided information, and verifying the proofs (signatures) of other players, as well as the modifications of the verified private state. This process may be costly, but note that a coordinator need not "replay" all of the game, but only a part (chosen at random). This may keep the cost low, while still deterring players from self-modification of private state.

The cost of maintenance of concealed or conditional state is highest in the initialization phase. This stage should be carried out only when an object is renewed. (when the object's state changes). The expense of this protocol may be controlled by reducing the constant number of object parts, at the cost of decreasing security. The number of object parts cannot be less than two. Table 4.1 shows a performance analysis of the most complex protocols in our procedural FM architecture: the protocols for management of concealed state. In the table, N is the number of objects (of concealed or conditional state); s - the number of shareholders; m*k - the number of replication players; and l - the number of objects stored by a peer. Note that computational cost of the distribution stage is complex - but this stage should be carried out only when an object is renewed. During most game operations, the cost of concealed state management is reasonable. The reconstruction phase has a constant (fetching the parts of an object).

All the proposed protocols have allowed us to realize one goal: limit the role of the central trusted component of the system (the coordinator). The coordinator does not have to maintain any state for the players. He participates in the game occasionally, during distribution of concealed/conditional state and during verification of private state. The maintenance of public state remains distributed, although it requires a higher communication overhead.

Prototype P2P MMO Game

The proposed architecture has been implemented in a prototype P2P MMO game. We have chosen this approach, rather than simulation, since we wanted to evaluate architecture in a more realistic setting. Implementing the proposed mechanisms in a real game has the advantage that all practical implementation issues need to be worked out, rather than implementing a more abstract version of the protocols in a simulator. The disadvantage is a smaller scale of testing and a more complex performance evaluation. However, the chosen approach has resulted in several practical performance improvements of the proposed architecture.

The prototype P2P MMO game has been implemented using FreePastry and Scribe. The game could be run on mobile phones that accessed the

Table 4.1 Complexity of concealed state management

	COMMUNICATION	COMPUTATIONAL	MEMORY
DISTRIBUTION STAGE	$O(N*(s+m*k))$ SIMPLE	1. secret sharing: a. matrix inversion mod p, $N*(k+1)$ times b. matrix multiplication, $N*(k+1)$ times; 2. hashing $N*(s+m*k)$ times COMPLEX	set of objects of concealed or conditional public state: $O(N)$ SIMPLE
RECONSTRUCTION STAGE	$O(s)$ SIMPLE	1. secret sharing: a. matrix inversion mod p b. matrix multiplication 2. hashing (s times) SIMPLE	1. public knowledge (hashes, replica shares location): $O(N*s+m*k)$ 2. private knowledge (secret shares, own objects): $O(N+l)$ 3. access history: $O(s*l)$ COMPLEX

Internet by Bluetooth. Since it turned out that running FreePastry on mobile phones was impossible due to the limited memory and processor speed, peers on mobile phones were connected to a single FreePastry node running on a Bluetooth access point with dual stack that was also connected to the Internet. Thus the network infrastructure was not a true ad-hoc Bluetooth network, but a hybrid network of Bluetooth and fixed links. The nodes running on Bluetooth access points acted on behalf of peers running on mobile phones.

As described in the architecture of P2P MMO games, the game should be divided into regions, and all players in a region constituted a group and joined a single Scribe topic. Each group has a common address prefix, allowing nodes of the group to be close to each other in the Pastry overlay. Each group had one trusted superpeer that could have different functions. Depending on configuration, the superpeer could be responsible only for the verification of private states of players, or additionally for the management of all public object in the game.

The P2P MMO game had no concealed or conditional public state. Location information was considered public state. All players managed their own private state and submitted to verification every time they needed to interact with a public object or with another player (instead of periodic verification - this form of verification is stricter than that proposed in the original

architecture). Public objects were managed by chosen peers - either by the superpeer, or by peers that were chosen by the superpeer and explicitly indicated to other peers in the group. This approach allowed for a transfer of trust from the superpeer to the manager of public objects and also limited the impact of attacks on overlay routing, since the chosen manager had to sign all of his messages using public key cryptography, and his public key was signed by the superpeer during the trust transfer. The implemented prototype allowed for performance evaluation of the proposed mechanisms for private and public state management in the procedural FM architecture.

For evaluation purposes, an automatic game playing agent was implemented that could move at random and interact with objects or other players. For simplification, the game was modified to allow the automatic agent to access all objects and all other players, regardless of his location (otherwise, it would be necessary to implement complex movement algorithms, which was not necessary for performance evaluation).

Performance Evaluation of Prototype Game

Performance evaluation was carried out by running multiple automatic game playing agents on single hosts. A total of 11 hosts was used in the experiment, one of which was used to run the superpeer. The hosts all had 512 MB of main memory and 1 GHz or 1.8 GHz processors. All hosts were connected by a 100 Mbit Fast Ethernet local network. Since the purpose of the performance test was to evaluate the overhead of the implemented procedural FM mechanisms, a simulation of a WAN was not required.

The experiment focused on a comparison of performance of the P2P MMO game with and without trust enforcement. Trust enforcement could be turned off, and then all private state modifications were executed by peers without verification, and all public object accesses and peer interactions proceeded unconditionally. The results of this part of the experiment were considered as a reference point for comparison with the same simulation with trust enforcement mechanisms turned on.

Only the performance results of the game in a stable state, with churn near to zero (about 1 percent), was considered in the evaluation. One of the reasons for this decision was that FreePastry and Scribe turned out to be very unstable under higher churn. Another reason was that with low churn, the reference performance of the game without procedural FM improved and stabilized, and therefore the evaluation of the performance overhead of procedural FM was more reliable.

For the purpose of increasing the number of nodes, multiple game agents were run on single hosts. However, here the scalability of the experiment was limited by FreePastry and Scribe. It turned out that it was not possible to run many nodes of FreePastry and Scribe on a single computer (regardless of the operating system, although performance was slightly better on Windows than on Linux). The maximum working number of nodes on a single host was about

50, however this amount of nodes resulted in very unstable performance. Therefore, in the experiment the number of game agents on a single host was in the range of 10-40, thus limiting the total number of nodes in the game group to about 400. Therefore, the experiment was run with the following group sizes: 100, 200 and 400 peers.

For the evaluation experiment, one region was created in the game, and this region had one coordinator (trusted superpeer). The superpeer was responsible for verification of private state. Every verification request to the superpeer consisted of the following operations:

- Routing the request to the superpeer through the overlay.
- Finding the client in a data structure.
- Verifying the clients' private state.
- Creating and signing the verification certificate.

Public state was managed by the superpeer or by peers chosen by the superpeer (who could change his decision over time, thus distributing the load between peers or allowing for peer departures). Regardless of who managed the public state, every request for public state access resulted in the following operations:

- Routing the request to the manager through the overlay.
- Finding the object in a data structure.
- Checking the signature on the verification certificate.
- Checking the position of the accessing player (for evaluation, this stage was modified so that it always allowed access. However, the same amount of operations was carried out as in the normal game.)
- Determining the results of the interaction.

These two types of requests, the verification request and interaction request, have been used to evaluate the performance of the procedural FM architecture. As described above, the trust enforcement mechanisms could be turned off. The experiment used the times of game operations with trust enforcement mechanisms as a reference value for the comparison with the game operation times that included time for procedural FM mechanisms of the verification and interaction requests. The time overhead is of crucial importance as a performance criterion in a game, since the game was interactive, and increasing the time of executing game operations could lead to poor game quality.

Figures 4.9 and 4.10 show the times of game operations that required private state verification or public object interaction, respectively. The two types of bars on the figures correspond to running the tests with trust enforcement turned off (black bar labeled "NT", short for "No Trust") and with trust enforcement running (gray bar labeled "TE", short for "Trust Enforcement"). 5 types of tests are shown on the figures. The first parameter of each of these tests is the size of the game group: 100, 200 or 400 peers. The second

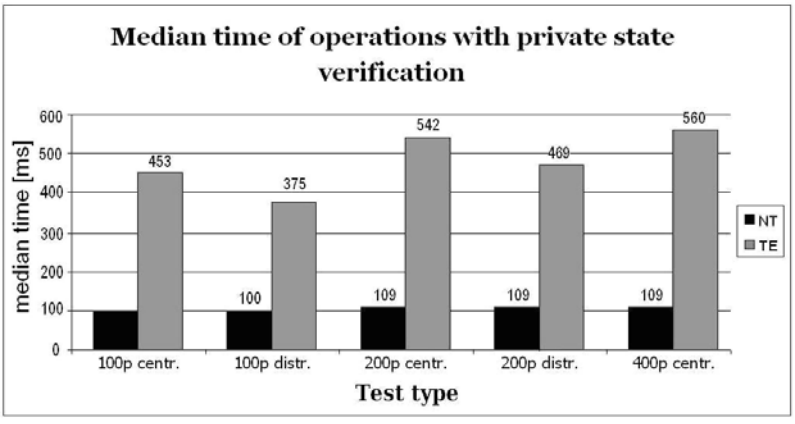


Fig. 4.9 Median time of game operations that require private state verification

parameter controls whether public state was managed by the superpeer (centralized) or distributed to other peers.

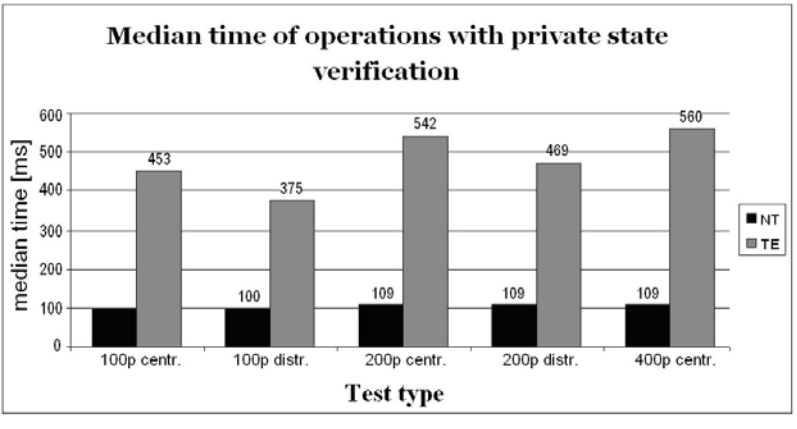


Fig. 4.10 Median time of game operations that require public state interaction

The results show that the delay increase due to using trust enforcement is significant, but for most cases, the game response times are within 1 second. This is quite an acceptable delay in MMO games. Note that in a Wide Area Network, the networking delays could well dominate over the delay increase due to using cryptography in trust enforcement.

Since the verification and interaction requests use Pastry overlay routing, the delay increase due to the use of trust enforcement increases with the growth of the game group size. It can be noticed that this increase is roughly logarithmic. The median time increases by about 100 milliseconds for

exponential increases of the group size. This corresponds to the known theoretical results concerning Pastry routing.

Distributing the functions of public state management from the superpeer to other peers has a twofold effect. Firstly, the median time required for public state interaction requests (when trust enforcement is used) increases by 200-300 milliseconds. This is due to the fact that the superpeer is usually better connected in the overlay than other peers, and the access time of other peers can be longer. On the other hand, the median time required for private state verification decreases by 70-80 milliseconds. This is due to the fact that the superpeer has a smaller computational load and can handle verification requests faster.

The results presented on Figure 4.9 and 4.10 showed the median response times. The distribution of response times resembles a heavy-tailed distribution. Some example distributions are shown on Figure 4.11. The Figure shows the times of game operations that required private state verification when trust enforcement was turned on, and management of public state was done by the superpeer. Distributions of times for three different group sizes are shown; it can be seen that increasing the group size increases the response time, but that this effect is limited and the increase from 200 to 400 peers does not significantly alter the distribution of response times. This confirms the theoretical properties of communication in DHT-based overlays. It can be seen, however, that increasing the group size leads to a narrowing of the distribution around its median value, which can be interpreted as follows: the delay for overlay routing begins to dominate in the response time over the other costs.

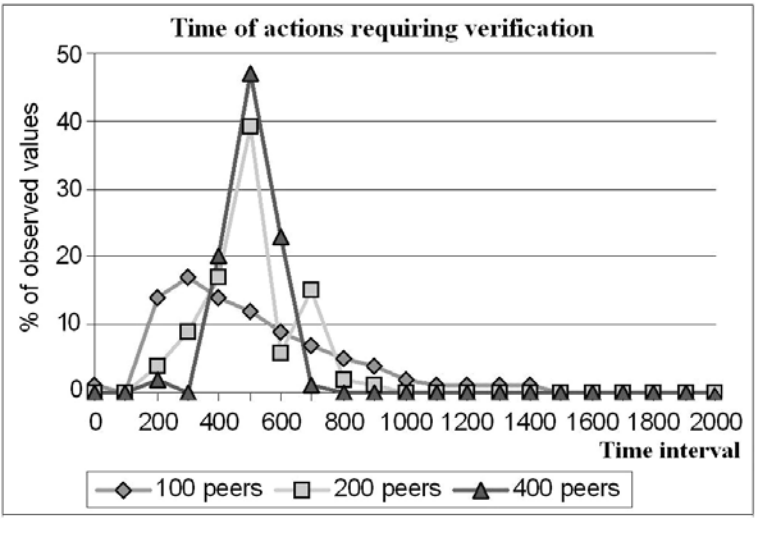


Fig. 4.11 Impact of group size on distribution of times of actions requiring verification

Distributing public state management from the superpeer reduces the number of requests for public state interaction at the superpeer. This effect is shown on Figure 4.12. The Figure shows the percentage of all interaction requests that are served by the superpeer. The line close to 100% is obtained when the public state is managed exclusively by the superpeer. (The line is not equal to 100% because some of the interaction requests are lost - probably due to the high load of the superpeer that could refuse to process a request.) The lower line is obtained when public state is distributed among peers (the superpeer participates in sharing the load). It can be seen that the distribution is effective and obtains fair load balancing regardless of the group size. There were a total of 11 peers (including the superpeer) that could be selected as managers of public state. Therefore, the percentage of interaction requests that was observed at the superpeer - about 9% - reflects a good quality of load balancing. A similar load was observed on other peers.

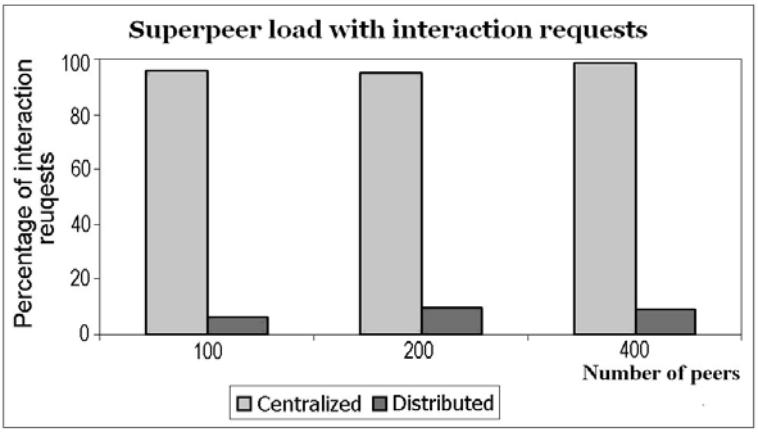


Fig. 4.12 Percentage of interaction requests served by superpeer

However, note that keeping public state management at the superpeer reduces the need for transferring trust to other peers.

Summary of Performance Evaluation

The practical implementation of the proposed procedural FM framework has demonstrated that it is possible to use procedural fairness enforcement in real P2P MMO games. The observed delays of game actions due to the use of trust enforcement were within 1 second for group sizes of up to 400 peers. While these are small groups (it was not possible to simulate larger ones with the available equipment, due to performance limitations of FreePastry),

the observed increase of delays was logarithmic, indicating that (if the trend observed for group sizes of 100, 200 and 400 were to continue), a group of the order of magnitude of 10000 should not observe an increase in game action times due to trust enforcement that would be larger than 2 seconds. This is still a delay that could be tolerated by players. Note also that the results of performance evaluation indicate that as group sizes increase, the time needed for overlay routing begins to dominate over the delays due to the use of cryptography in trust enforcement.

The experiments confirmed that the distribution of public state management from the superpeer to other peers is necessary and can achieve fair load balancing. Note that we have not used the overlay in our implementation of the load balancing, but allowed the superpeer to choose peers responsible for public state management and to explicitly transfer trust to those peers by notifying the game group. This approach solved several security problems, and allowed for load balancing that was not vulnerable to hot spots in the DHT.

4.2.1.7 Summary of FM in P2P MMO Games

Many applications that could benefit from the P2P computing model cannot use it because of concerns over security and fairness. In this section, we have attempted to show how a very sensitive application (a P2P Massive Multiplayer Online game) may be protected from unfair user behavior. However, let us note that while our research focused on P2P MMO games as a challenging application, the developed procedural FM protocols could be applied in other P2P applications. An example could be P2P auctions that have many properties similar to the discussed conditional or concealed game state in MMO games. Future work may consider applying trust enforcement mechanisms in P2P e-commerce applications.

In our proposed procedural FM architecture, we have been forced to abandon the pure peer-to-peer approach for a hybrid approach (or an approach with superpeers). However, we have attempted to minimize the role of the centralized trusted components. The result is a system that, in our opinion, preserves much of the performance benefits of the P2P approach, as exemplified by the P2P platform for MMO games proposed in [80]. At the same time, it is much more secure than the basic P2P platform.

We see trust as yet another security functionality (such as privacy or authentication) that can be provided by known cryptographic primitives. The approach that we have tried to use for procedural FM in peer-to-peer games is considerably different from previous work on trust management in P2P computing, that has usually relied on reputation. However, reputation systems are vulnerable to first time cheating, and are difficult to use in P2P computing because peers have to compute reputation on the basis of incomplete information (unless the reputation is maintained by superpeers). Instead, we

have attempted to use cryptographic primitives to ensure a detection of unfair behavior and to enable trust.

The proposed trust management architecture was validated by an implementation in a practical P2P MMO game. The performance of the implementation was tested and it was found that the use of trust enforcement incurs a practical performance overhead. Therefore, the use of the proposed trust enforcement mechanisms is indeed practical in P2P games. As the sizes of game groups increases, the delays due to overlay routing begin to dominate over additional delays due to the use of cryptography.

The mechanisms that form the proposed trust management architecture work on a periodic or irregular basis (like periodic verification of private players by the coordinator or Byzantine agreement after a veto). Also, the possibility of cheating is not excluded, but rather the trust enforcement mechanisms aim to detect cheating and punish the cheating player by excluding him from the game. In some cases, cheating may still not be detected (if the verification, as proposed, is done on a random basis); however, we believe that the existence of trust enforcement mechanisms may be sufficient to deter players from cheating and to enable trust, like (usually) in the real-world case of law enforcement.

4.2.2 Distributed Queues

Queues, especially priority queues, are one of the principal mechanisms for ensuring procedural fairness. In an ODS, queues may often be used by individual agents; however, in this case they will always rely solely on local knowledge. In order to apply a queue as a mechanism for assuring procedural or distributional fairness in an entire ODS, usually a centralized element of control is required. In this centralized controller (which can be an agent in the ODS that is trusted by all others), a single queue (or a system of queues) is created that has access to global information. Such an approach, as any other that relies on a centralized control element, is vulnerable to failures and scalability problems, as well as imposing an additional requirement of a-priori, unconditional trust of all agents in the controller.

It is possible to avoid the difficulties of using a centralized controller in an ODS and still to make use of a queue that possesses global knowledge. This approach can be called distributed queuing. In this section, we shall describe how a distributed priority queue can be created using a distributed sorting algorithm. The proposed algorithm is based on gossiping protocols and ant algorithms.

A key challenge in the design of a distributed queue for ODS is the need to rely on local information (avoiding centralized control). However, this approach makes the distributed queue vulnerable to cheating by adversaries who can attempt to improve their position in the queue (or worsen the position of their competitors). A distributed queue may not be able to prevent

such cheating strategies using only local information, but it should at least be able to detect cheating adversaries. Therefore, the protocol of the distributed queue (based on a distributed sorting algorithm [64]) should be supplemented by appropriate cryptographic mechanisms. It turns out that the use of agents who are trusted a priori cannot be altogether avoided; however, their functionality can be reduced, and the system does not need to rely on a single such agent.

The priorities used to control the distributed queue can be any value (depending on application semantics); in the systems discussed here, this value is a measure of global trustworthiness. This measure is used to create a priority queue where the most trusted agents are first.

We shall assume the existence of a standard TMS that calculates trust or reputation of a given agent in a given context. In a distributed system, such a service can be implemented using one of the many proposed variations of Peer-to-Peer trust management systems, such as Eigentrust [69], or using a structured P2P network. We also assume that TMS is capable of calculating trust that is objective. This does not mean that the service does not calculate subjective trust, but only that it can additionally give a trust value for any user that does not depend on the trustor (e.g. by employing WOWA [33]).

Of course, the distribution of control creates security vulnerabilities. Adversaries may be interested to be first in the priority queue, although their priorities do not satisfy the selection requirements. Our goal is therefore *ensuring fair ordering of agents*, in the presence of adversaries. To this end, we use a special protocol design and cryptographic primitives. We assume that there are several trusted components (superpeers) in our network which control the fairness of all agents.

4.2.2.1 System Organization

We consider an ODS of N agents. We model the distributed system as a general graph, where the semantics of the link is that of a logical connection that exists if two agents interact with each other. Initially, an agent n_i connecting to the network connects to d other agents, called neighbors (d is a parameter called the *network degree*). However, the system is highly dynamic. During experiments we allow agents to disconnect and join the system according to a churn factor.

Each agent in our system can communicate only with its own neighborhood (set of its neighbors). A single agent has no knowledge about the system size and existing connections besides its own neighborhood.

We assume a strong authentication mechanism, that makes whitewashing and creating "virtual" peers (Sybil attacks) impossible. This can be easily accomplished by a distributed Public Key Infrastructure (PKI) that uses a group key to sign public keys of joining peers [1].

We assume that there are several trusted components (superpeers) in our network called arbiters, which control the fairness of all agents [172]. Arbiters can be selected from all agents (for example 2% of most trusted agents will be the arbiters).

Relationship to Trust Management Service

The proposed system can be used as a priority queue that uses any information to determine the priority. However, we propose to apply this system to determine the most globally trusted agents in the ODS.

For the selection of trusted agents in a specific context, trust can be an application-specific value. We distinguish between trust and reputation, but the selection protocols could work using any of the two concepts. All that is required is that trust or reputation would be expressed on an ordinal scale, enabling direct comparison. Since trust and reputation are context-specific, the selection must also be in a specific context. From now on, we shall assume that such a context is given, although our protocols can work with many contexts at the same time.

The selection of agents by trust from a global population requires that trust values should be available and comparable for all agents. We here assume that there is a Trust Management Service (TMS) capable of providing such trust values. TMS takes as an input the ID of any agent n_i and a context C, and returns trust value t_i^C. Note that t_i^C is independent of the agent who executes the service.

The TM service is also capable of accepting reports in a specified context, in other words, of receiving information that would be used to calculate reputation in a context. This capability will be used by our protocol as a way of providing incentives for fair protocol execution. The TMS itself will be used to control the fairness of execution of the priority queue protocol.

4.2.2.2 Sorting of Agents by Priority

A queue must realize the following main functions: *push*, which adds a new agent into the queue with a specified priority, and *pop*, which removes the agent with the highest current priority from the queue. However, before we describe how these functions can be implemented, we will explain how the distributed priority queue maintains its most important invariant: that nodes in the queue should be sorted according to their priority (or, in our case, their global trust).

Existing centralized sorting algorithms are expensive and vulnerable to bottlenecks, moreover they are not suitable to maintain actual ranking in the face of churn and the large scale of our network. In our approach we use a gossip-based sorting algorithm which is a modification of [64]. We extend the original algorithm by secure initial random number generation, by validating

the swap requests and by proposing the different types of ants that perform these swaps.

In the beginning of the algorithm, each agent n_i is assigned a random number r_i. r_i values are chosen with a uniform distribution from $[0, 1]$. During the algorithm computation, each agent swaps its r_i with nodes from its neighborhood according to the value of their priority. With the progress of the algorithm, the order of r_i gradually starts to reflect the order of the priorities. The detailed description of the algorithm follows.

Random number generation is computed using one of the secret sharing methods. When agent n_i starts the algorithm, n_i requests all its neighbors to generate the pre-shared secret. Each neighbor n_j of n_i generates a pre-shared secret P_{s_j}, signs it and sends it back to n_i. n_i computes the random number r_i using all the pre-shared secrets, using bitwise XOR operation (or similar) $r_i = P_{s_1} \oplus P_{s_2} \oplus P_{s_3} \ldots \oplus \ldots P_{s_k}$. n_i keeps the pre-shared secrets as proofs.

The algorithm proceeds by exchanging random *swap requests*. In each swap request, a requesting agent (n_i) sends a *message* to a randomly chosen agent n_j. The responding agent n_j compares the priorities (trust values) t_i^C and t_j^C. If the order of t_i^C and t_j^C is different than the order of random values r_i and r_j (i.e. $t_i^C < t_j^C$ and $r_i > r_j$, or $t_i^C > t_j^C$ and $r_i < r_j$), the two agents swap their random values. We assume that both the request and the reply are signed by both agents, regardless of the result. After the exchange each agent sends the result to its arbiter. The arbiter can verify the correctness of the agents' r_i and r_j values with a certain probability that depends on the trust the arbiter has in the agents (in the context of protocol execution).

During the swap request, the manner in which agent n_i chooses its exchange partner n_j turns out to be crucial for the algorithm's performance. In the original algorithm [64], the exchange partner n_j was picked out at random from the whole population. In our version of the algorithm, we also consider local exchanges (only within n_i neighborhood) and k-step random walks (different possibilities are detailed in the experimental section).

After a number of such swap requests, using r_i, agent n_i can estimate the relation of its priority t_i^C to priorities of other nodes in the whole population. Assume that the initial random numbers are chosen with an uniform distribution from $[0, 1]$. If we want to select the 10% most trusted agents, we simply take the agents which have $r_i > 0.9$. An agent that receives a high r_i and keeps this value for l subsequent exchange rounds (l subsequent swap requests) is deemed to be located at the beginning of the priority queue. When the *pop* operation is performed on the queue, such agents (usually a very small number, perhaps only one) will reply to the *pop* request. A final comparison of their priorities can be then carried out to determine the correct order of leaving the priority queue.

The algorithm does not stop, as we assume that both the network and the priorities (trust values) t_i^C of agents can change. However, a service (such as the *pop* operation of the distributed priority queue) that uses the algorithm's results will need them at a specific moment (even though the result might not

reflect the recent changes). Consequently, we will evaluate the order proposed by the algorithm not only in steady-state (after the sorting stabilizes), but also during execution.

4.2.2.3 Evaluation of the Distributed Priority Queue's Invariant

We evaluate the algorithm presented in the previous section from two different perspectives. Section 4.1 considers procedural fairness, i.e. how the algorithm works in the presence of malicious adversaries that try to improve their position in the priority queue. Section 4.2 evaluates the performance of the sorting algorithm, measured as the quality of proposed ordering.

Procedural Fairness of the Priority Queue

In this section, we will present how the protocol achieves procedural fairness of results in the presence of adversaries. We shall therefore focus on adversaries that attack our protocol for the purpose of leaving the priority queue, even though their priorities are too low. We do not consider the possibility of falsifying priorities by the agents but assume that these priorities are globally known and can be verified (for our example application, the priorities are global trust values calculated by the TMS; we do not consider the possibility of attacks on the TMS service).

To influence the outcome of protocol operation in their favor, adversaries can:

1. Choose values for initial r_i in a non-random manner. If many adversaries choose the same random number, the protocol will not be able to sort by exchanging these numbers.
2. Cheat in exchanges of random numbers by claiming a higher priority t_i^C.
3. Cheat in exchanges of random numbers by altering r_i in between exchanges (for example using its r_i from previous swap).
4. Cheat at the end of sorting by announcing false r_i.

Note that we use the assumption that agents cannot create "virtual" agents (using the Sybil attack), nor can they change their identity (strong authentication).

We will discuss how the algorithm copes with these problems in subsequent paragraphs.

Initial random numbers. As mentioned in Section 3, if the initial r_i is generated by an agent independently of its neighborhood, it is possible that all r_i are not sufficiently varied. Thus, the resulting order cannot reflect the order of priorities.

Consider a situation, in which a group of agents chooses the maximum number r. That may create regions in the network where sorting will be handicapped or even impossible. In such regions, the algorithm cannot swap r_i, as all the values are equal.

We cope with this situation by controlling the way in which initial r_i are generated. A node computes its r_i using secret sharing. The parts of the secret are obtained from the random values given by the node's neighborhood. Moreover, n_i keeps these values as proofs (the values are signed by the neighbors who sent them).

Claiming higher priority. Adversaries can be completely prevented from cheating during the exchange of random numbers if, every time an exchange is made, both parties summon an arbiter (possibly more than one) who calculates the trust values of both parties and returns a verdict. However, calling a trusted third party for each exchange would be too expensive. Therefore, we use a random verification that depends on the reputation of an agent in the context of fair sorting. If the agent's reputation is below a threshold, his decisions to swap random numbers will be checked more frequently. Setting this threshold to a high value ensures that agents are strongly discouraged from cheating. Of course, any verification of the swapping requires an arbiter that can be any randomly selected agent. Finally, if the swapping decision of an agent is incorrect, a negative report in the context of sorting fairness will be passed to the TM service.

Altering r_i. An adversary can choose an arbitrary value of r_i in between swaps. However, in the following swap, such an arbitrary value will not have a correct signature. Recall that we request that r_i is signed by agents who generate a random value, and then it is signed in each swap. Therefore, a false r_i can be easily detected in the next swap. Upon detecting such behavior, the detecting agent fills a report that reduces the cheater's reputation (such a report can be easily validated by checking the r_i and the false signatures). Note that this schema requires that the sorting algorithm does not stop. In our algorithm, agents check their trust values randomly. Therefore, in a longer period of time, any false value of r_i will be detected.

Performance of Distributed Priority Queue

Algorithm Implementation. We have implemented our sorting algorithm using the *Anthill* framework. Swap requests and resulting r_i values are represented by so-called ants. An ant uses a modified random walk with x steps, or the gossiping algorithm [64]. We tested two types of random walks, performed by ants and a randomized swap between all the nodes in the network:

- RWxAnt - A standard random walk with range x. The ant contains a swap request. If the swap method cannot be executed by peer n he must send the request to his neighbor. Each time an agent forwards the swap request, the value of x is decreased by 1, and when it reaches 0 the request is sent back to the requesting agent. For instance, if $x = 1$, every agent in the network can swap its random value r_i only with its neighborhood. In the worst case, each ant generates $2 * x$ messages, as each ant returns to the requesting peer using exactly the same path it used in the forward walk.

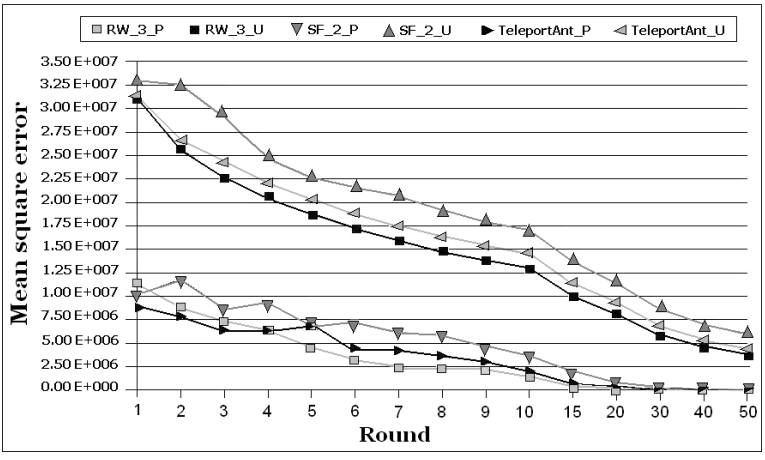


Fig. 4.13 Effect of distribution of trust value on quality of sorting

- SFxAnt - Sniffing First with x sniffing steps. Here, an ant should visit x agents and check their t_i^C and r_i values. Among these x agents, the ant computes the best candidate for the swap operation, taking into account the priority t_j^C and the random number r_i. Each request in this algorithm always generates $2 * x$ messages.
- TeleportAnt - Swap requests are performed between random agents in the network, without considering neighborhoods [64].

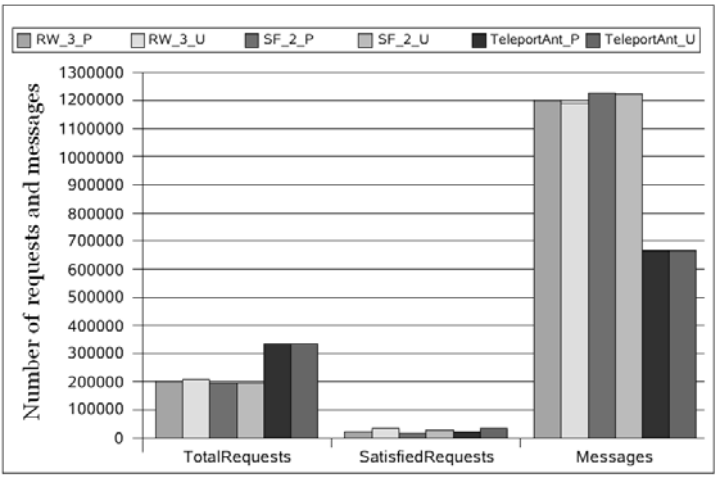


Fig. 4.14 Effect of distribution of trust value on traffic generated by algorithm

Testing parameters. We run our sorting algorithms several times with different parameters that are: network size, network degree, probability distribution of trust values t_i^C, and churn factor. Each ant type was tested with a network of 10, 100, 1000, 10000 and 100000 nodes. The network degree parameter (d) defines the initial number of connections of every node in the network. We have set d to 2, 3, 4, 5, 7 or 10. We tested two probability distributions of priorities t_i^C: uniform and Pareto. We also tested dynamically changing networks with a 5% churn factor (constant during single experiment). Churn has been implemented as random arrivals and departures of similar numbers of nodes (the average size of the network does not change).

Experiments. Each experiment was repeated 10 times, and we present average results. We have calculated t-Student 95% confidence intervals for the evaluation metrics, but the width of the confidence interval was usually about 1% of the value. For that reason, the confidence intervals are not visible in the presented figures.

A single experiment consists of 50 rounds. In the beginning of each round, each agent generates an ant with a probability 0.5 and a timeout set to $2ttl+1$, where ttl is the amount of iterations that ant can exist in the network (the agents do not generate a new ant until the ttl drops to 0 or its ant comes back). After that, the ants wander through the network. At the end of the round, we gather the current state of sorting and evaluate its quality using one of the proposed metrics.

Evaluation metrics. The main metric used to evaluate the results of sorting is the mean square error of the current ordering, relatively to the perfect ordering. Let us assume a static snapshot of the network after t rounds of the algorithm. Without loss of generality we assume, that the agents are numbered according to the non-decreasing t_i^C values: $t_i^C \leq t_{i+1}^C$. n_1 is the agent with the least priority, and n_N is the agent with the highest priority.

Let us denote as p_i the rank of agent n_i, taking into account only the ordering by its current r_i value. For instance, if $p_i = 1$, n_i has the lowest r_i in the system.

Our quality measure is the mean square distance between the ordering resulting from r_i values and the proper ordering:

$$Q_t = \sum_i (p_i - i)^2.$$

This measure converges to 0 when an algorithm sorts the priorities t_i^C completely.

In Figure 4.15, we present the effect of the initial distribution of priorities on the quality of sorting. Note that the distribution of priorities can be quite uneven, if they depend on agent properties that have an uneven distribution in the ODS, such as, for example, availability of peers in a P2P system. We run 6 scenarios using RW_3, SF_2 and $TeleportAnt$ assuming uniformly distributed priorities, and the same ants assuming a priority distributed in agreement

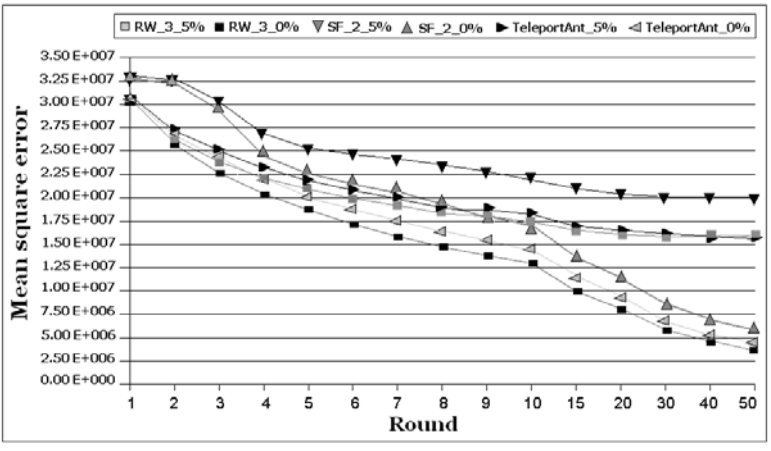


Fig. 4.15 Impact of churn on quality of sorting

with the Pareto distribution. We gathered results after 50 rounds. The quality of sorting improves linearly with the course the algorithm, but it does not converge to perfect order during the simulation time.

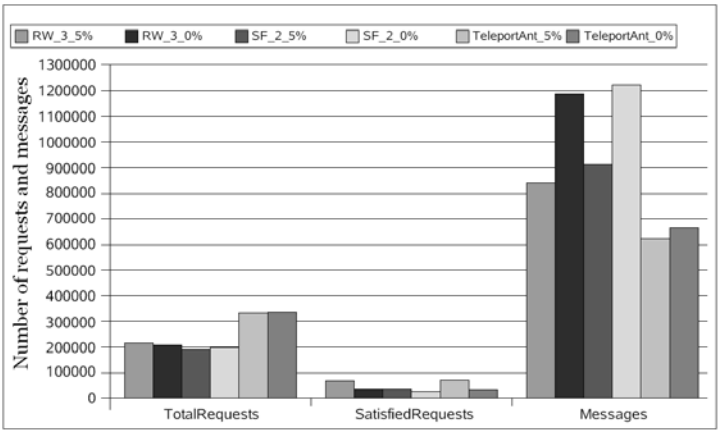


Fig. 4.16 Impact of churn on traffic generated by algorithms

For both uniform and Pareto distributions, the best results were achieved by the random walk ant with range $x = 3$. The result of *TeleportAnt* was almost as good as the best ant. The sniffing ant algorithm with range $x = 2$ was little worse than the others ants, but generated less swap requests than the *TeleportAnt*. As shown in Figure 4.16 the message overhead for *TeleporAnt* was much lower than for the other ants. This is because the

longer range ants wander through the network for a long period of time before they come back to the requesting agent.

The *TeleportAnt* seems to offer the best price-performance for both distributions, but we should take into consideration that this ant uses another algorithm to select the group of agents to swap with. Another drawback is overheads connected with routing that *TeleportAnt* algorithm omits (ant virtually "teleports" to the destination). This indicates that the simple random walk is a robust choice for various distribution types.

Note that on average, using RW_3 the number of swap requests is about 20.6 per agent during 50 rounds. This means that in our application that determines the most trusted agents, on average, an agent must call the TMS less than once in every two turns.

In the next experiment we tested the impact of churn on sorting quality. We ran 6 scenarios using RW_3, SF_2 and *TeleportAnt* (the same as in the previous scenario) assuming uniformly distributed test values. We have tested these ants with no churn (0%) and with churn factor 5% (constant a during single experiment). Churn has been implemented as random arrivals and departures of similar numbers of agents (the average size of the network does not change). Generally, the quality of sorting improves linearly with the algorithm's operation time, and similarly to the first experiment, the results after 50 rounds do not converge to perfect order. As predicted, convergence is much slower for the churn factor at 5%. After 50 rounds, the best results for the churn factor 5% were achieved by the TeleportAnt and random walk. The results are almost identical, but the Telepor-tAnt achieves these results by generating more swap requests. The sniffing ant algorithm with range $x = 2$ algorithm is considerably worse than the others.

As in previous experiment we must take into the consideration that *TeleportAnt* uses another algorithm, which helps to select the best candidates to swap. Generally a random walk algorithm with step $x = 3$ is well suited for dynamic environments.

4.2.2.4 Summary of Distributed Queues

Priority queues are widely used to achieve procedural fairness. Distributed priority queues can be especially useful in open, fully distributed systems, where central control is hard to achieve. In this section, we have proposed a protocol that implements a distributed priority queue in a distributed manner. The proposed protocol requires the existence of trusted third parties (the arbiters who can evaluate the ordering in the priority queue). However, the role of these arbiters can be reduced if for any decision, more than one arbiter is called, and the arbiters make a decision based on the majority of opinions.

The *pop* operation of the proposed priority queue can be implemented by any selected agent that sends a *pop* message to all agents in the ODS (using an application-layer multicast protocol, this can be achieved efficiently). When

an agent receives the *pop* request, it evaluates the condition described above: whether or not it has kept its r_i value for the recent subsequent l swap rounds (l is a system-wide parameter that can be set to an appropriately large number). Such agents respond to the originator of the *pop* request specifying their priority values. The agent which executes the *pop* operation can order the responding agents by their priorities, if she receives more than one response.

The *push* operation of the priority queue is performed simply by a node that joins the network and starts to exchange its r_i value. Such a node will eventually be placed in the right position in the priority queue.

An example of an application of a distributed priority queue can be the selection of superpeers in a P2P overlay network. Nodes that are chosen for superpeers must be the most trusted of all. Furthermore, our algorithm can ensure that a constant percentage (for example, 10%) of the most trusted nodes will be superpeers at any given time.

We have tested the fairness and performance of the distributed priority queue. Its fairness depended on the resilience of the sorting algorithm to adversaries that tried to modify the algorithm's operation. We used simple cryptographic measures such as secret sharing and digital signatures to increase the security of our protocol. While cheating is not made impossible by these measures, it becomes possible to detect unfair behavior. The protocol uses the available trust management service to maintain a reputation of agents in the context of fair sorting of trust values.

Finally, we investigated the performance of various kinds of sorting algorithms. For this purpose, we simulated gossiping protocols that used various kinds of localized random walk. Of these, the most resilient to the distribution of trust values and to churn was a simple random walk. The range of this walk should depend on the level of churn; for more dynamic systems, higher ranges of the random walk are preferred. During tests, we have also detected that our algorithm is scalable. Results for network size $n = 10$ are similar to $n = 100000$.

Using our protocol, it is possible to determine the most trusted agents in a distributed, efficient and fair manner. The cost of the protocol lies in a frequent execution of the trust management service during sorting, that is made necessary due to the dynamic nature of trust. Other types of priorities may be less dynamic.

4.2.3 Distributed Agreement

An important type of fair procedure is an agreement (consensus) or voting procedure. Essentially, the task is to determine a choice of an action that is approved by a required set of agents in the ODS (this required set can be a simple majority, all agents, or some other set of usually more than 50% of agents). In distributed systems, this problem has been considered since the 1980s when Lamport defined the so-called Byzantine agreement problem.

This problem can be defined as follows: how should the agents in an ODS be able to agree on an action (or value) in the presence of adversaries who participate in the system and cannot be distinguished from other agents? Note that in the case of Byzantine agreement, disagreeing agents are considered adversaries. These protocols could tolerate a certain number of adversaries (if there are t adversaries, the early protocols usually worked with at least $2t + 1$ fair agents, but this was due to an assumption of a simpler failure model where nodes could only silently leave the system). However, for a long time, Byzantine agreement protocols have been considered as too costly to use in practice, due to a high (usually quadratic) communication overhead. Another reason for this opinion was that previous Byzantine agreement protocols required unrealistic assumptions about low network delays (synchronous operation) and simple failures of nodes (they did not consider malicious adversaries).

Recent research on Byzantine agreement has resulted in new, promising protocols that can be considered practical given today's computing and communication infrastructure. One such protocol was PBFT [28] that used Byzantine agreement to synchronize a set of replicas in an ODS. PBFT relies on a coordinator (the primary replica) to control the agreement protocol (any agent can be a coordinator for a certain period of time). Also, PBFT uses a three-phase commit protocol in order to write new data to the replicas. PBFT guarantees safety (replica consistency) and liveness (the ability of processing new requests by the system) as long as there are at most t adversaries and $3t + 1$ fair agents in the ODS.

Another protocol, Paxos [110] reduced the 3-phase commit protocol of PBFT to a 2-phase commit, thus reducing the communication overhead of the protocol. Zyzzyva [84] makes use of the agents that consume the replicated content in order to support the agreement process (the previous protocols used the replicating agents themselves). Bosco [153] is a protocol that can achieve Byzantine agreement in one communication step at the cost of tolerating fewer adversaries (Bosco requires $7t + 1$ fair agents for strong consistency guarantees).

Quorum-based systems such as Q/U [3] are another form of distributed agreement protocol that offers weaker guarantees and assumes static membership. An operation in a quorum-based system is successful if it can achieve a quorum (find a required number of agents that can offer a service). Quorum-based systems are also tolerant to Byzantine failures (malicious adversaries), but only if the number of adversaries is smaller (Q/U requires $5t + 1$ fair agents).

Practical applications such as Chubby (Google) or Zookeeper (Yahoo) use Byzantine agreement to provide fault tolerance. Byzantine agreement also has applications in P2P systems. Hybrid approaches such as hierarchical Byzantine agreement protocols offer an interesting and intuitive way of reducing communication costs and increasing fault-tolerance [189]. Introducing trusted

agents into the ODS or using a trust management system directly is a promising direction for improving distributed agreement protocols.

4.3 Emergence of Fairness as a Result of Trust Management

In this section, we return to the subject of chapter 3 and investigate the relations between trust and fairness in ODS. There exists much evidence to support the idea that the two concepts are related; a connection exists even at the level of basic, abstract definitions of trust and fairness, as discussed in section 2.2.1.1. Studies of indirect reciprocity demonstrate another example of a deep and complex relationship between a special kind of trust management mechanism (reputation) and a special kind of fairness (indirect reciprocity) (see section 2.3.7). In this section, we shall investigate the relation between trust and fairness more directly, using the definitions and measure of distributive fairness developed in section 2.3.

Ensuring fairness of resource distributions in ODS is difficult. On the other hand, trust management (TM) is widely used to improve procedural fairness. It therefore seems tempting to consider TM methods as a tool to improve distributional fairness, as well.

The question considered in this section is whether or not TM systems can also be used to ensure or increase fairness of resource distribution. While this is different (and more difficult) from procedural fairness, the two concepts are related. Norms and rules of behavior are often defined with the fairness of resource or cost distribution in mind. As an example, consider the laws that oblige all citizens to pay taxes. Enforcing procedural fairness (abiding by the tax laws) has the goal of enabling efficient resource redistribution by the government (among its other duties). Such a resource redistribution should result in increased fairness of income distribution.

The question considered in this section can be formulated as the following hypothesis: in successful reputation (or trust management) systems, fairness should be an emergent property. By emergence we understand the arising of a complex property (fairness) out of simpler system behavior (the use of a reputation system by agents). The emergent behavior should also be qualitatively different and not reducible to the sum of simpler behaviors. This is the case of emergence of distributive fairness due to trust management, because the increase of distributional fairness is qualitatively different from even very high mutual trust of agents. We shall refer to this hypothesis as the Fairness Emergence (FE) hypothesis. In this section, the FE hypothesis has been verified.

We will use a simulation approach to verify the FE hypothesis. However, our goal is not just to see whether the hypothesis applies in an abstract model, but to verify the validity of the Fairness Emergence hypothesis in realistic conditions. In order to realize this goal, we need to study the behavior of a

popular, well understood trust management system. The natural candidate for such a system is the reputation system used by Internet auctions. Previous studies have established that the use of reputation systems increases the total utility of agents [134, 142], and investigated the sensitivity or reputation systems to selfish or malicious user behavior [34]. This study investigates how the use of reputation impacts the fairness of the distribution of agents' utilities.

The goal of verifying the Fairness Emergence hypothesis under realistic conditions can be fulfilled by a study of Internet auction systems under non-stationary conditions, and in the presence of selfish and malicious users. Reputation systems used in other applications, such as P2P networks, are vulnerable to the same effects. Therefore, our model of a reputation system is sufficiently general to apply to different applications, while at the same time we are able to draw on the well-known properties of reputation systems used in Internet auctions in order to increase the realism of our model. This is done at the risk of drawing conclusions that will apply mostly to Internet auctions. The realism of our study of reputation systems for Internet auctions is increased further by the use of trace-driven simulation (to our knowledge, this is the first such study described in the literature. We have obtained a large trace from a Polish Internet auction provider that is used in the second group of simulations to realistically model agent presence in the system. However, the results from our first group of simulations are sufficiently general to warrant drawing conclusions about the FE hypothesis in the domain of other applications.

Also, the fairness of distributions of users' utilities in Internet auctions is an important goal in its own right. Buyers or sellers in Internet auctions expect that if they behave as fairly as their competitors, they should have a similarly high reputation. In other words, the users of a reputation system expect that the reputation system give a fair distribution of reputations. In the absence of other differentiating factors, this should also ensure a fair distribution of utilities. This expectation of users is a consequence of the general social norm: people expect fair treatment from many social and business institutions, like a stock exchange, or an Internet auction site.

The questions considered in this section are therefore the following: is the FE hypothesis universally true? Does the FE hypothesis apply under realistic conditions? How sensitive is fairness emergence to the performance of a TM system? What are the conditions that can lead to a lack of fairness emergence due to the use of a TM system? Does fairness emergence occur if agents are infrequently unfair? Does fairness emergence occur if agents have a low sensitivity to reputation? Does fairness emergence occur if agents employ discrimination? These and similar questions can lead to a better understanding of the ability of trust management systems to increase fairness of distribution of costs or resources in an open, distributed system without central control.

4.3.1 Considering Fairness in Trust Management Systems

In most research, a trust management system is considered successful when the sum of utilities of all agents in the ODS is highest (recall this property of the Prisoner's Dilemma that was introduced in section 2.3.5.1). Note that the utilitarian paradigm is used even if the experiment or simulation uses a more complex model of agent interaction than the Prisoner's Dilemma.

The use of Prisoner's Dilemma allows for an implicit consideration of agent fairness, while the sum of utilities is considered explicitly. Yet, in a more realistic setting, the assumptions of the Prisoner's Dilemma may not be satisfied, and it is possible to point out cases where the utilitarian approach fails to ensure fairness: in an online auction system, a minority of agents can be constantly cheated, while the sum of utilities remains high. A notable example of explicit consideration for fairness of reputation systems is the work of Dellarocas [34]. An attempt to demonstrate that explicit consideration of fairness leads to different results in the design and evaluation of reputation systems has been made in [178].

In a real-world setting, users of trust management systems would be expected to have quite varied levels of utility (perhaps even incomparable ones). How, then, do we expect a trust management system to realize a goal of fairness? And how can the Fairness Emergence hypothesis be true?

This concern is based on a frequent misconception that mistakes equality for fairness. If a trader in an Internet auction house has better goods, provides better services and has better marketing than other traders, it is perfectly fair that he should have a larger transaction volume and a larger revenue. In fact, his reputation should increase, as well, so the trust management system should in this case support him in getting even more trade. On the other hand, if we have two honest traders that have comparable goods, services, and marketing, yet they have very unequal reputation and transaction volumes, surely something is wrong in the way the trust management system works.

Therefore, *when all other factors can be excluded* (equivalent to the ceteris paribus assumption from economics), fairness can be identified with distributional fairness. In a laboratory setting, such conditions can be satisfied and we can design trust management systems that realize the goal of fairness, even in the presence of adversaries.

4.3.2 Verifying the Fairness Emergence Hypothesis by Simulation

To verify the Fairness Emergence hypothesis, we used simulation experiments. The FE hypothesis would hold if we could establish that the reputation system causes an increase of the equity of the distribution of utilities. In

particular, we will be interested to study the impact of the quality of the reputation system on the equity of utility distributions.

The simulator is based on the Repast 3.1 platform [138] and resembles an Internet auction system. In the design of the simulator, we had to make a decision about a sufficiently realistic, yet not too complex model of the auction system, of user behavior, and of the reputation system. We chose to simulate the reputation system and the behavior of its users as faithfully as possible (the only simplification is that we use only positive and negative fe edback).

The auction system, on the other hand, has been simplified. We simulate the selection of users using random choice of a set of potential sellers. The choosing user (the buyer) selects one of the sellers that has the highest reputation in the set.

After the buyer has selected a seller, a transaction between the two agents may occur. However, this is not always the case in our simulations, because the chosen seller may have a reputation that is too low for the buyer. If the chosen seller has a reputation below the buyer's acceptance threshold, no transaction will occur. Still, we count the number of such *transaction attempts* in our simulation. The number of transaction attempts is used as a measure of time in our simulation (since we assume that each transaction attempt would consume some time and effort on behalf of a buyer, the number of such transaction attempts is limited). Furthermore, the granularity of transaction attempts in our simulation is very high. To show meaningful results, we group several hundred subsequent transaction attempts into one *turn*. The turn is used as a measure of time for the demonstration of simulation results.

In this paper, we describe two sets of simulation results. The first set was obtained from simulations of a closed system of agents – the set of agents was kept fixed for the duration of the simulation. This approach was used initially to reduce the number of factors that could impact the results, and to study fairness emergence in a simpler setting. The second set of simulation results was obtained from a trace-driven simulation approach that was used to control the presence of sellers in the system. This allowed for a more realistic simulation of an open system of buyers and sellers, where the time a buyer or seller could spend in the system was controlled by the trace. The second set of simulation results takes into account more complex factors, but was used to verify the results from the first set.

4.3.2.1 Agent Behavior

In our simulator, a number of agents interact with each other. There are two types of agents in the system: fair and unfair agents. Dishonest agents model adversaries. To test the FE hypothesis, we shall be interested in the fairness of utility distributions of fair agents. The payoffs of fair and unfair agents will also be compared.

In the closed system simulations, all agents are similar. In the trace-driven simulations that will be discussed in more detail below, agents can be buyers or sellers (this separation is a consequence of the separation of roles in real auction systems, where users mostly either buy or sell). This additional distinction makes the simulations more realistic.

When an agent wants to carry out a transaction, it must make three decisions. The first decision concerns the choice of a transaction partner (seller) and whether or not to engage in the transaction. The agent chooses his partner from a randomly selected set of k other agents (in the simulations of the closed system, k has been equal to 3 or 1). From this set, the agent with the highest reputation is chosen. However, if the highest reputation is lower than a threshold p_{min}^{choice} (in the closed system simulations, fair agents choose partners with a reputation of at least 0.45, and unfair agents: 0.3), then the choosing agent will not engage in any transaction. If the best agent's reputation is sufficiently high, the choosing agent will engage in the transaction with a certain probability p (in the simulations presented here, this probability was 1).

The second decision concerns the agent's behavior in the transaction. This decision can be based on a game strategy that can take into consideration the agent's own reputation as well as the reputation of his partner, the transaction history and other information. We decided to use the famous Tit-for-tat strategy developed by Rapaport but extended it using a reputation threshold: if two agents meet for the first time and the second agents' reputation is below p_{min}^{game} , the first agent defects. The strategy used in the simulations presented here is also based on the threshold p_{min}^{cheat}. In the case where the partner's reputation is higher than p_{min}^{cheat}, the agent would act fairly; otherwise, it would cheat with a certain probability c. In the simulations presented here, fair agents had a cheating probability of 0, while unfair agents had a cheating probability of 0.2 and a reputation threshold of 0 - meaning that unfair agents cheated randomly with a probability of 0.2.

The third decision of the agent concerns the sending of reports. For positive and negative reports, an agent has separate probabilities of sending the report. In the simulations presented here, the probability of sending a positive report, p_{rep}^{+} was 1.0, while the probability of sending a negative report p_{rep}^{-} varied from 0 to 1. This choice is based on the fact that in commonly used reputation systems [179], the frequency of positive reports is usually much higher than of negative reports. In the simulation it is also possible to specify a number of agents that never send reports. This behavior is independent of the honesty or dishonesty of agents.

The strategies of agents in our simulations do not evolve, but remain fixed for the duration of simulation. In this respect our work is different from the research on evolution of cooperation or indirect reciprocity [182, 165]. Our research is focused on verifying the effect of trust management on fairness, without considering how the strategy of using trust management or reputation has evolved – that is the concern of related and future work [134, 27].

Note here that the presented model of agent behavior with respect to the reputation system matches many kinds of applications. The model has been described using Internet auctions as an example. Another kind of realistic application is a Peer-to-Peer system. A transaction in such a system is an exchange of data or services (resources). Unfair behavior in such a system is called free-riding: peers use resources of others, but do not reciprocate. A P2P application can use reputation to combat free-riding. The reputation system in a P2P application is distributed, in contrast to the reputation system used in Internet auctions. However, the discovery of proofs by the P2P reputation system is affected by the quality of the distributed search algorithms and by the presence of adversaries, who can attempt to drop negative proofs. A result is a smaller availability of negative reports, which has been modeled in the simulator by varying the probability p^{-}_{rep} varied from 0 to 1. This type of adversary has been discussed frequently in the literature [99, 88, 69]. The first set of our simulations presents results that can also apply to P2P applications that use a reputation system.

4.3.2.2 Reputation System Warmup

A real reputation system has a large initial history that can be used to evaluate infrequently present agents. In the simulation approach, this initial history had to be reproduced. In the closed system, for each simulation, the first 20 turns have been used to warm-up the reputation system by acquiring an initial history of agent behavior. This means that the payoffs have note been recorded, but an agents' reputation has been modified by positive and negative reports. This method has been used to model the behavior of a real reputation system, where the system has available a long history of transactions. Simulating the reputation system without a warm-up stage would therefore be unrealistic.

In a closed system, it is possible to warm-up the reputation of all agents at the same time, at the beginning of the simulation. In the open, trace-driven approach, the trace represents a period of time taken from the operation of a real Internet auction site. Agents present in the trace could have been present in the system before the beginning of the trace. As this information is not available, it is also not realistic to simulate the system without a warm-up. However, this warm-up can be done separately for each seller. If a buyer would select a seller that was in the warm-up stage, the results of the transaction were not recorded in the utility of the buyer and the seller. The seller's reputation was updated. A fixed number of l transactions was used as a warm-up. This ensured that the reputation system had some initial information about each seller, before the buyers utilities were recorded. In the simulation results presented below, $l = 5$. Reducing the length of the warm-up had a strong effect on emergence: emergence was not observed for $l = 0$, for any other setting of simulation parameters.

4.3.3 Fairness Emergence in a Closed System

4.3.3.1 Experiment Setup

In simulations of the closed system, there was a total of 1500 agents, out of which 1050 where fair and 450 were unfair. While the proportion of unfair agents is high, they cheat randomly and at a low probability - so a unfair agent is really a "not totally fair agent". Also, considering that frauds in Internet auctions are among the most frequent digital crimes today, and considering that cheating in a transaction may be more frequent than outright fraud - it may be sending goods that are of worse quality than advertised - this proportion of unfair agents seems realistic.

The simulator can compute reputations using all available feedbacks. The results of the simulation include: the reputations of individual agents and the total utilities (payoffs from all transactions) of every agent. In the simulations presented here, an agent's reputation is computed as the proportion of the number of positive reports about the agent to the number of all reports.

All simulations were made using pseudo-random numbers, therefore the Monte Carlo method has been used to validate statistical significance. For each setting of the simulation parameters, 50 repeated runs were made, and the presented results are the averages and 95% confidence intervals for every calculated criterion. The confidence intervals were calculated using the t-Student distribution.

We decided to use transaction attempts instead of the number of successful transaction as a stop condition because we believe that an agent would consider each transaction attempt as an expense, and the reputation system would have to work well after as few transaction attempts as possible. In most presented simulations for each turn, 500 transaction attempts have been made.

For each simulation, the first 20 turns have been used to warm-up the reputation system. It means that the payoffs are not recorded but an agents' reputation is modified by positive and negative reports. This method has been used to model the behavior of a real reputation system, where the system has the availability of a long history of transactions. Simulating the reputation system without a warm-up stage would therefore be unrealistic.

4.3.3.2 Closed System Simulation Results

To verify the Fairness Emergence hypothesis, we have been interested to investigate the impact of a reputation system on the equity of the agent utility distribution. Equity of utility distributions was measured using a fairness criterion based on the theory of equitable optimality: the area below the Generalized Lorenz curve (BLC). However, other criteria such as the sum of agent

utilities are considered as well. The simulations revealed that the Fairness Emergence hypothesis holds in several cases, but not universally; therefore, we have investigated the sensitivity of fairness emergence to various factors that influence the quality of the reputation system.

4.3.3.3 Fairness Emergence in the Long Term

The first studied effect was the emergence of fairness in the long term. In the simulation experiment, we measured the area under the Generalized Lorenz curve (BLC) and ran the simulation until the BLC stabilized. This experiment was repeated using three scenarios: in the first one, the agents did not use any reputation system, but selected partners for transactions randomly. In the second experiment, the reputation system was used, but agents submitted negative reports with the probability of 0.2. In the third experiment, negative reports were always submitted.

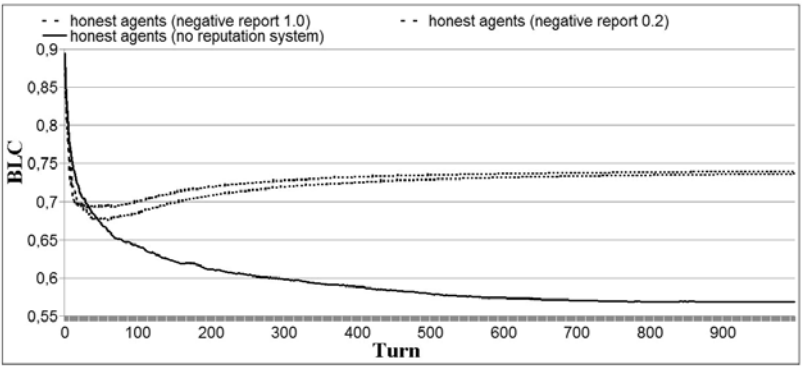


Fig. 4.17 Fairness Emergence in the long term

The results of the three experiments are shown in Figure 4.17. The Figure plots the average BLC of fair agents from 50 simulation runs against the number of turns of the simulation. It can be seen that when agents do not use the reputation system, the BLC stabilizes for a value that is almost twice smaller than the value of BLC that is obtained when reputation is used. Furthermore, there is a clear effect of increasing the frequency of negative feedback: the BLC increases faster and stabilizes at a higher value when $p^{-}_{rep} = 1$. The initial decrease of the BLC from 1 is due to the fact that at the beginning of the simulation, the distribution of fair agent utilities is equal (during the warm-up stage, utilities of agents are not recorded. All agents start with a zero utility after warm-up is completed.)

The result of this experiment seems to be a confirmation of the FE hypothesis. The distributions of fair agents' utilities have a higher BLC (and a higher total sum of utilities) when the reputation system is used. Yet, the problem here is that in realistic auction systems, most agents only have a small number of successful transactions, because they use the system infrequently. In our simulation, new agents did not join the system (although the number of agents was large). The average number of successful transactions of an agent was about 270, which is much lower than the number of agents; this means that as in a real auction system, the chance of repeated encounters was low. However, this number is still large. The simulations were continued until a stable state was reached; in practical reputation systems, such a situation would not be likely to occur because of the influx of new agents and the inactivity of old ones. For this reason, we have decided to investigate the FE hypothesis in the short term, or in unstable system states.

4.3.3.4 Fairness Emergence in the Short Term

The simulation experiments used to study short-term system behavior have been about 8 times shorter than the long-term experiments. For these experiments, the number of successful transactions of an average agent was about 60. Figure 4.18 shows the BLC of the distributions of fair agents' utilities. On the x axis, we show the number of turns. The figure shows two lines corresponding to different frequencies of sending negative reports by fair agents (unfair agents always sent negative reports). The results show that for low negative report frequencies fairness emerges more slowly. Increasing the available negative reports reduces the time needed for fairness emergence. This effect is apparent very quickly, even after 50 turns of simulation.

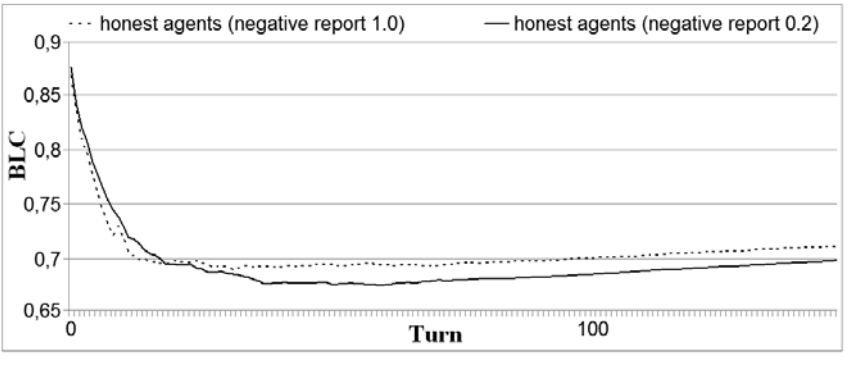


Fig. 4.18 Fairness Emergence in the short term

From now on, fairness emergence in the short term will be studied more closely to verify whether the improvement of reputation system quality will strengthen fairness emergence. In other words, until now we considered fairness emergence with time, and now we shall consider the *sensitivity of fairness emergence to the reputation system's quality*. All further experiments have been made in the short term, outside of the stable state of the system.

4.3.3.5 Effect of Better Usage of Reputation

The usage of reputation by agents had a particularly strong influence on the emergence of fairness. In our simulations, during a transaction attempt, agents chose a seller with the highest reputation from a set of k candidates. The chosen candidate needed to have a reputation that was higher than the buyer's threshold. If $k = 1$, then the transaction partner was chosen at random and only the threshold p_{min}^{game} was used to consider reputation. If $k = 3$, it was less likely that an agent with a lower reputation would be chosen as a transaction partner. These two scenarios correspond to the real life situation of buyers who are able to select sellers from a larger set, based on their reputation; on the other hand, it could be possible that the choice is low, because only one seller has the required goods or services.

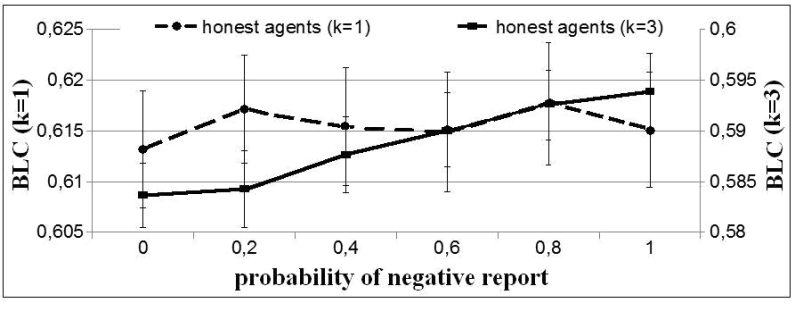


Fig. 4.19 Effect of increased choice on BLC

We considered the two scenarios while investigating the impact of the frequency of feedback on the reputation system. It turns out that increasing the choice of agents is necessary for the emergence of fairness. Figure 4.19 shows the effect of increasing the frequency of negative feedback on the BLC of fair agents. The figure shows two lines that correspond to the scenarios of $k = 1$ and $k = 3$. It can be seen that if the choice of agents on the basis of reputation is possible ($k = 3$), then the increase in the number of feedbacks leads to a increase of BLC. On the other hand, if the choice is limited ($k = 1$), then the increase in the level of negative feedbacks, does not have a statistically significant effect on the BLC. This effect is best explained by the fact that if

choice is available, honest agents have a better chance of avoiding dishonest agents while at the same time they do not waste transaction attempts. If agents do not have choice, they can still avoid transactions with dishonest agents, but they will waste transaction attempts and have a lower utility.

Figure 4.20 shows the effect of increased choice and varying negative feedback frequency on the sum of fair agents' utilities. It can be seen that once again, enabling the choice of partners based on reputation has a positive effect on the welfare of fair agents. For k=3, fair agents overall had a higher sum of utilities than for k=1, and this sum increased when the frequency of negative reports increased. This also explains why the Gini coefficient of fair agents for k=1 was lower than for k=3. Since the sum of utilities was lower for k=1, the Gini coefficient could also be lower, although this does not mean that the distribution of utilities for k=1 was more equitable than for k=3.

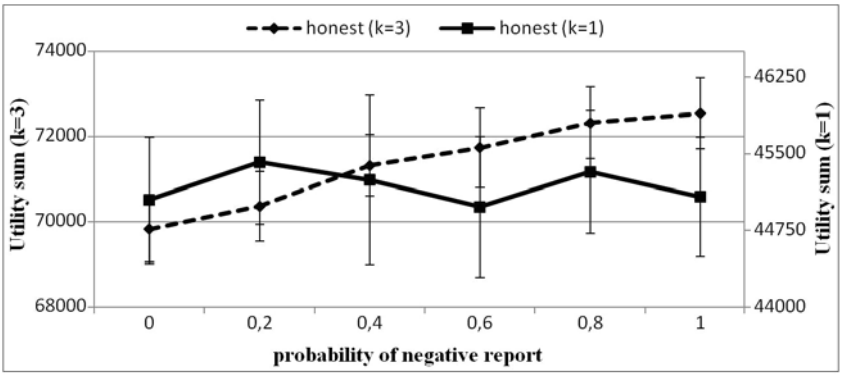


Fig. 4.20 Effect of increased choice on sum of utilities

4.3.3.6 Effect of Better Feedback

Better feedback is a prerequisite for increasing the quality of a reputation system. For that reason, we chose to investigate the effect of increased feedback on the emergence of fairness. As has been explained previously, the frequency of negative feedback was varied from 0 to 1. We also varied the frequency of positive and negative feedback simultaneously; however, for the simple reputation algorithms considered in this paper, the only significant parameter is the proportion of negative to all feedback. For that reason, varying negative feedback's sending frequency is sufficient to evaluate the system's sensitivity to feedback availability. Another issue related to feedback is the possibility that agents send false feedback. Our studies indicate that a small amount of false feedback does not impact the results, but a significant amount of false feedback will confuse any reputation system. For this reason, in this analysis we disregard the possibility of sending false feedback by the agents.

Figure 4.21 shows the effect of increasing negative feedback on the sum of utilities of all agents. It turns out that the total sum was not affected by the increase. This seems to be a paradox, since we are using the iterated Prisoner's Dilemma as a model of our auction system. In the Prisoner's Dilemma, increased fairness of agents results in an increased sum of all utilities. And increasing negative feedbacks from 0 to 1 should result in decreasing the ability of unfair agents to cheat.

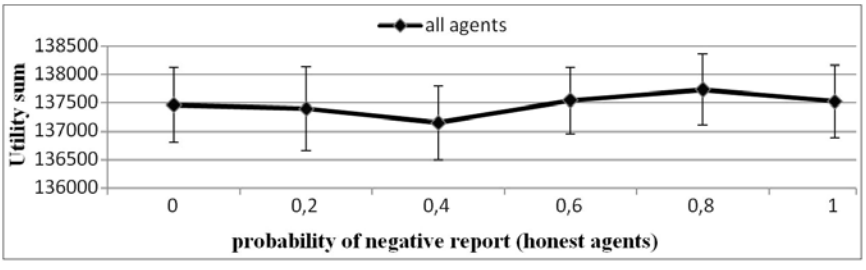


Fig. 4.21 Effect of increased feedback on sum of utilities of all agents

This experiment also shows that even assuming the use of a Prisoner's Dilemma as a model of a transaction, the use of the sum of all agents' utilities (the utilitarian paradigm) would lead to an erroneous conclusion that the system behavior is not affected. From the utilitarian point of view, the reputation system works equally well when the frequency of negative reports is 0, as when it is equal to 1.

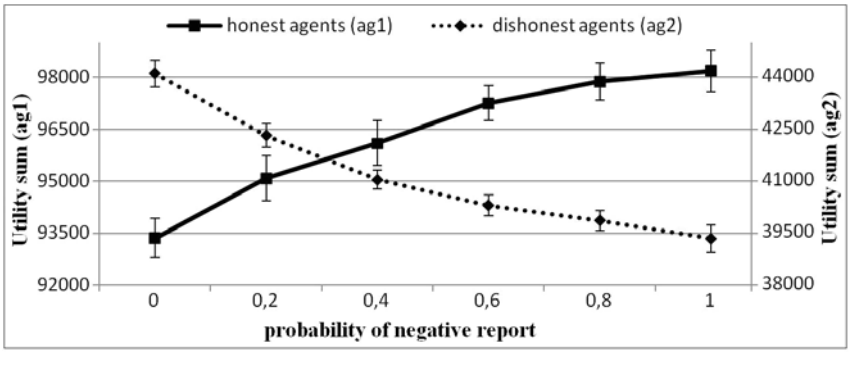


Fig. 4.22 Effect of increased feedback on fair and unfair agents' utilities

Figure 4.22 shows that this is not the case. The sum of utilities of fair agents increases, as negative feedback is sent more frequently. On the other hand, the sum of utilities of unfair agents drops. The reason for this fact is

that with higher frequencies of negative feedback, the reputations of unfair agents decrease, and therefore these agents have fewer successful transactions. On the other hand, fair agents manage to avoid unfair ones, and do not waste transaction attempts (therefore they have more successful transactions and higher payoffs in these transactions).

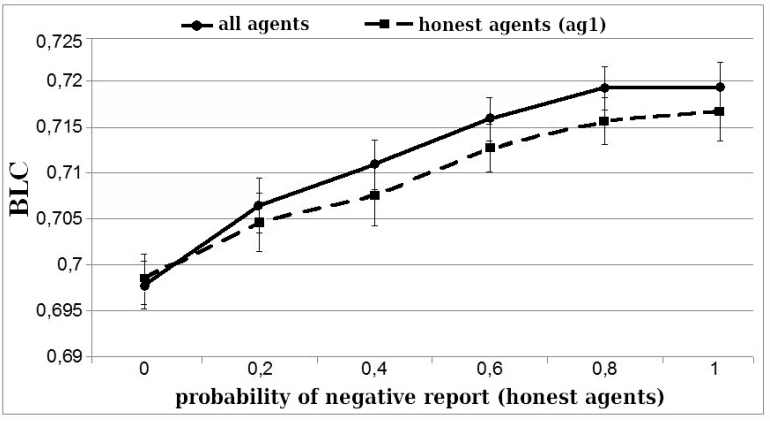


Fig. 4.23 Effect of increased feedback on BLC

Figure 4.23 shows the effect of increased negative feedback frequency on the BLC. Clearly, increased negative feedback frequency leads to an increased BLC of honest agents' utilities. Note that the effect is statistically significant for the variation of from 0 to 1 (also from 0.4 to 1). Note that these simulations have been made in the short term and that together with the results about the sum of utilities, they prove the FE hypothesis: increasing the quality of the reputation system does indeed lead to more equitable distribution of fair agents' utilities, as the hypothesis suggested.

4.3.3.7 Effect of Improved Reputation Algorithm

Another important type of fairness emergence could occur if the algorithm that is used to calculate reputations is changed. With better algorithms, perhaps it would be possible to improve fairness. That would be equivalent to fairness emergence with improved trust management system's operation.

4.3.3.8 Algorithm of Implicit Negative Feedback

The algorithm described in this section has been introduced [179]. Most online auction sites use a simple feedback-based reputation system [141]. Typically, parties involved in a transaction mutually post feedbacks after the transaction is committed. Each transaction can be judged as '*positive*', '*neutral*', or

'*negative*'. The reputation of a user is simply the number of distinct partners providing positive feedback minus the number of distinct partners providing negative feedback (possibly normalized by the number of all distinct partners). As pointed out in [104], such a simple reputation system suffers from numerous deficiencies, including the subjective nature of feedback and the lack of transactional and social contexts. Yet another drawback of feedback-based reputation systems is that these systems do not account for the psychological motivation of users. Many users refrain from posting a neutral or negative feedback in fear of retaliation, thus biasing the system into assigning overestimated reputation scores. This phenomenon is manifested by high asymmetry in feedbacks the collected after auctions and, equally importantly, by the high number of auctions with no feedback provided. Much of this of these missing feedback might convey implicit and unvoiced assessments of a poor seller's performance which should be included in the computation of a seller's reputation.

As described in [179], there can be many ways of identifying implicit feedback in a real-world reputation system, based on the observation of behavioral patterns. To evaluate the effectiveness of using implicit feedback, we have identified a simpler reputation algorithm that can be simulated and compared to the algorithm of most Internet auction houses.

Consider a user u with a history of n auctions. Let us assume that only $m \leq n$ of these auctions have feedback. Out of this m feedback m^+ are positive or neutral feedback (in practice, the amount of neutral feedback in reputation systems of online auctions can be ignored), m^- is negative feedback, while $m^* = n - m$ is the amount of missing feedback (transactions that had no feedback). Thus, $m^+ \leq m \leq n$. The reputation ρ_u of the user u will be calculated as follows:

$$\rho_u = \frac{m^+}{\alpha m^* + m^+ + m^-},$$

where $0 \leq \alpha \leq 1$. Thus, if $\alpha = 0$, the above reputation score becomes a simple ratio of the number of items of positive feedback received by the user u. In a case where the user has had no auctions, the above formula is undefined. In such a case we set the reputation ρ_u to an initial value, ρ_0. The coefficient α is used to control the importance of implicit negative feedback.

To be precise, in our simulations we use a slightly more complex version of the above algorithm. Since agents in the simulator choose whom they want to interact with on the basis of reputation scores, it is necessary to avoid the reputation dropping suddenly to a low level. This can happen in the initial phase of the simulation, when the reputation score has not yet stabilized (initially, a single negative feedback could decrease the initial reputation by a large degree). Therefore, we use a simple moving average to smooth reputation changes. The smoothed reputation $\rho_u^{ma}(t) = 0.5\rho_u^{ma}(t-1) + \rho_u(t)$, where t is time, and $\rho_u^{ma}(0) = \rho_0$(the smoothed reputation is initialized by

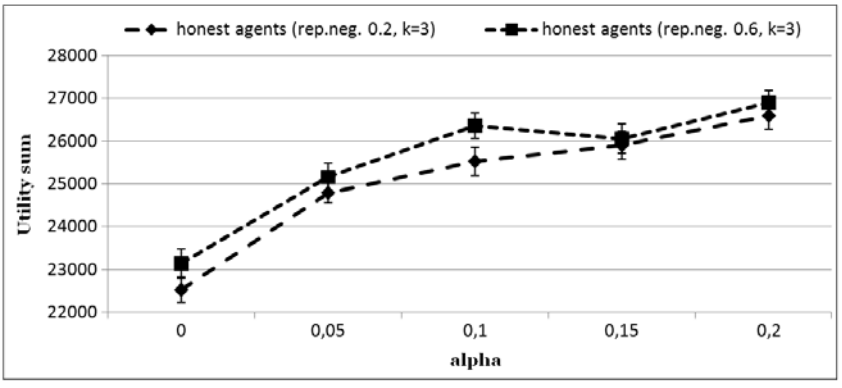


Fig. 4.24 Effect of improving reputation algorithm on utility sum

the initial reputation value). Note that over time, the impact of the initial reputation decreases exponentially.

The results of increasing α from 0 to 0.2 on the Gini coefficient of honest agents' utility distribution and on the sum of honest agents' utilities are shown in Figure 4.25 and 4.24, respectively. The figures show several lines that correspond to various frequencies of negative reports. Increasing the role of implicit negative feedback clearly increases fairness, and the effect is strong and statistically significant. This behavior is a confirmation of the FE hypothesis in an unstable state, in the presence of adversaries, and when the probability of negative reports is low. The sum of honest agents' utilities also increases when α is increased. Note that the effect of varying α from 0 to 0.2 is similar to the effect of increasing the probability of negative feedback.

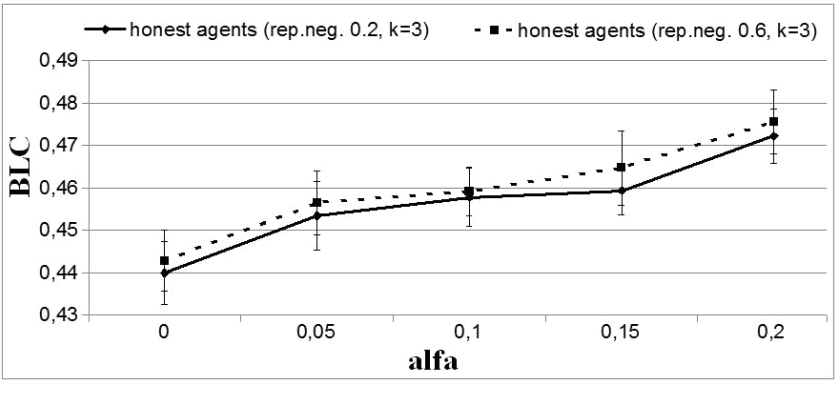


Fig. 4.25 Effect of improving reputation algorithm on BLC

Larger values of α have been found to lead to an increase of the Gini indicator in our previous research [178] but this effect was obtained for a different, specific version of the simulations system and needs to be studied further to allow generalization.

4.3.4 Trace-Driven Simulation of an Internet Auction System

In the previously described simulations scenarios, all agents were treated equally (although we have referred to the agent who initiated a transaction attempt as a "buyer", all agents had an equal chance to become a "buyer" in the described simulations). Moreover, all agents had a similar level of activity in the system. Agents could not leave the system during the simulation and were chosen over and over again for transaction attempts. New agents could not join the system. This approach had an impact on the evaluation of the reputation system. In a realistic reputation system, the amount of information available about new agents would be considerably less than the amount of information available about agents that have been active for some time in the system. In the closed system, in the long term, the reputation system would have very good information about agents. Considering the operation of the reputation system in the short term partially reduces that problem, but does not fully solve it.

The reason for the use of the closed system is that without additional information or assumptions, it was not possible to specify how active the agents should be in the system. In this section, we are going to remove this limitation. On the other hand, the previously described results were more general and could apply to a variety of applications of reputation systems.

We have obtained a trace from a large Polish Internet auction site. The trace includes approximately 200 000 seller transactions from 6 months. In the trace, there were about 10000 sellers randomly selected from the auction house. The weekly number of seller transactions in the trace is shown on Figure 4.26. The trace was used to control the times spent by sellers in the simulated system. In other words, using trace-driven simulations allowed us to simulate an open system. Figure 4.27 shows the distribution of the number of transactions made by a seller. It can be seen that this distribution resembles a heavy-tailed distribution.

The behavior of sellers was not recorded in the trace, and it is therefore simulated as described in the previous section. Moreover, the buyers are not trace-driven. Buyers initiate transactions with sellers who are present in the system at a given time (in the trace-driven simulations, one turn is equivalent to one day of the trace. During this turn, only the buyers who offered auctions on that day are present in the system). Buyers choose sellers in the same way as in the simulations of the closed system, choosing a seller with the highest

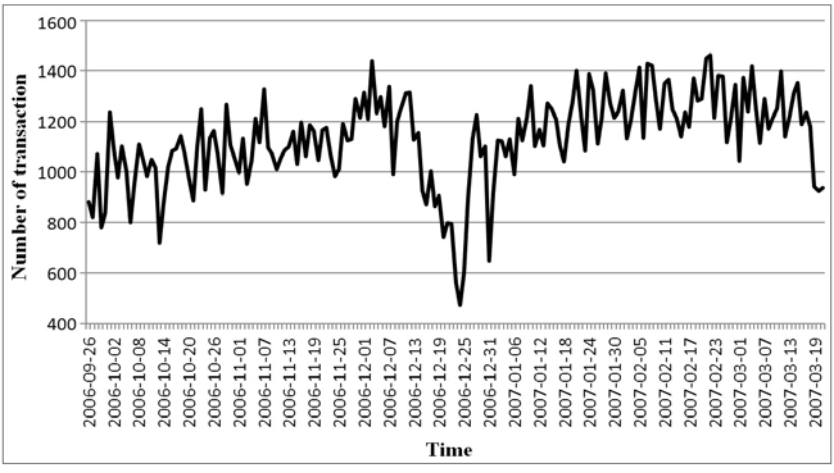


Fig. 4.26 Daily number of seller transactions in the trace

reputation from a random set. Buyers are also able to reject transactions if a chosen seller's reputation is below a threshold.

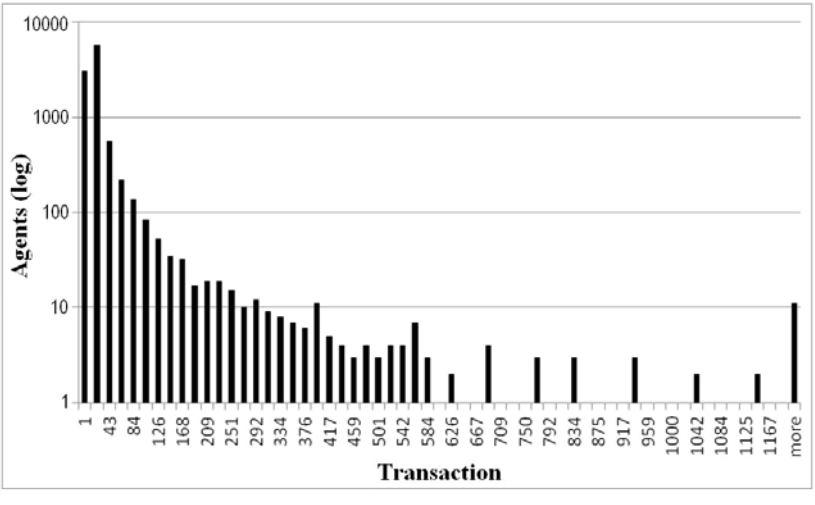


Fig. 4.27 Distribution of number of transactions of a seller

The simulations also used a different kind of warm-up. In the closed system, it was possible to warm-up the reputation of all agents at the same time, at the beginning of the simulation. In the open, trace-driven approach, the trace represents a period of time taken from the operation of a real Internet auction

site. Agents present in the trace could have been present in the system before the beginning of the trace. As this information is not available, it is not realistic to simulate the system without a warm-up. This warm-up needed to be done separately for each seller. If a buyer selected a seller that was in the warm-up stage, the results of the transaction were not recorded in the utility of the buyer and the seller. The seller's reputation was updated. A fixed number of l transactions was used as a warm-up. This ensured that the reputation system had some initial information about each seller, before the buyers utilities were recorded. In the simulation results presented below, $l = 5$. Reducing the length of the warm-up had a strong effect on emergence: emergence was not observed for $l = 0$, or for any other setting of simulation parameters.

4.3.5 Fairness Emergence in the Open System

To test the FE hypothesis in an open system, we have measured the utilities of two kinds of agents: the buyers and the fair sellers. By the FE hypothesis, the distribution of both kinds of utilities should become more equitable with an improvement of the reputation system. The results of the experiments partially support this hypothesis: the distributions of buyers become more equitable, but the BLC of the distributions of fair sellers does not vary significantly. We attribute this result to the chosen simulation scenario. Varying the probability of negative reports had an impact on a seller's reputation, but we have not simulated unfair buyers, so no effect on the utilities of fair sellers has been observed.

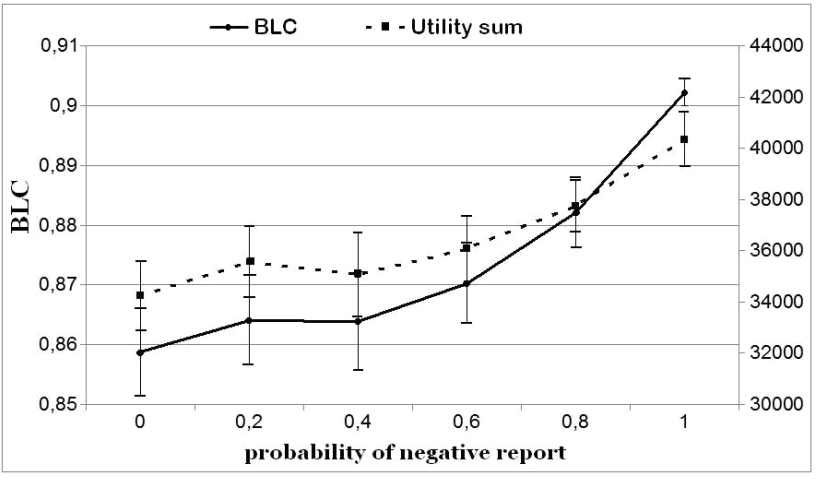


Fig. 4.28 Fairness emergence in the open system

Figure 4.28 shows the effect of increasing the probability of negative reports on the sum of utilities of all buyers and on the Generalized Lorenz curve of the distribution of buyers' utilities. Since both BLC and the utility sum increase, it can be concluded that the distribution of buyer's reputation indeed becomes more equitable. The effect becomes statistically significant for an increase of the probability of negative reports from 0 to 0.8. Thus, the Fairness Emergence hypothesis is partially confirmed in a realistic open system.

The reputation system effectively prevents unfair sellers from exploiting buyers. Figure 4.29 shows the sum of utilities of all unfair sellers that decreases with the increasing probability of negative reports. Once again, the effect becomes statistically significant for an increase of the probability of negative reports from 0 to 0.8.

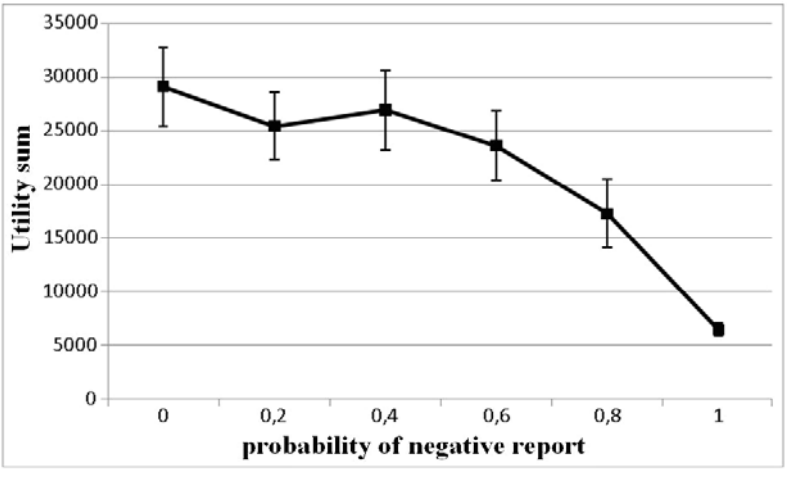


Fig. 4.29 Utilities of unfair sellers in the open system

We have investigated the sensitivity of Fairness Emergence to the behavior of unfair sellers (adversaries). To observe this effect, the probability of cheating by a unfair seller was varied. The results are shown in Figure 4.30. As expected, the Fairness Emergence was strongest in the case of unfair sellers cheating with probability 1. This meant that the reputation system could easily spot adversaries. Decreasing the probability of cheating weakens Fairness Emergence, with the values of 0.5 as a threshold. For lower probabilities of cheating, Fairness Emergence was not observed. This result can be explained by the simple reputation algorithm used in our simulation scenario (recall that reputation is a simple ratio of the number of fair transactions to the number of all transactions).

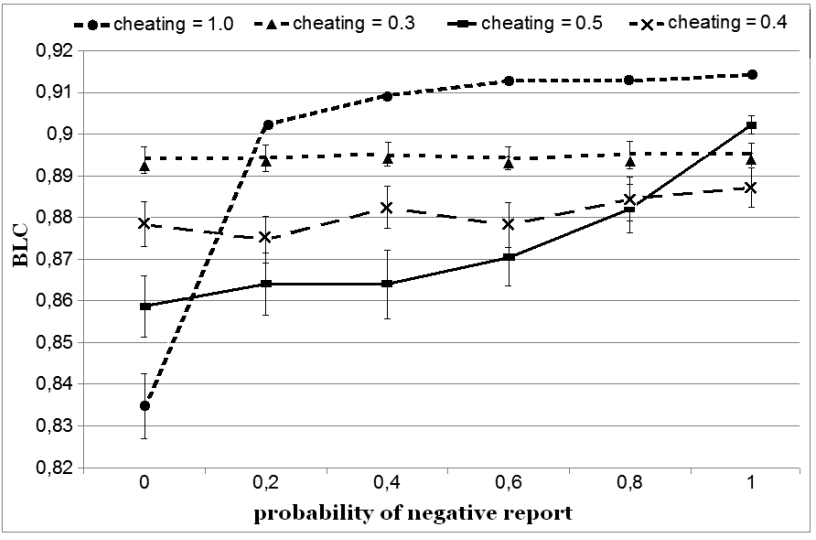


Fig. 4.30 Sensitivity of Fairness Emergence to unfair seller behavior

4.3.6 Consequences of Fairness Emergence

We have identified the crucial conditions for Fairness Emergence due to the use of a Trust Management system to occur. These are:

- the possibility of choosing the agent to interact with on the basis of reputation or trust,
- the availability of sufficient correct proofs (in the tested scenarios, reports for the reputation system),
- the use of sufficiently good trust management algorithms (in the tested scenarios, algorithms for computing reputation).

The Fairness Emergence hypothesis and the well-known "evolution of cooperation" are both examples of spontaneous emergence of a more complex behavior due to simpler behaviors of agents. It is noteworthy that trust also plays a part in the evolution of cooperation, because the use of reputation information supports direct and indirect reciprocity. Fairness Emergence also has a theoretical importance for the social sciences, because it demonstrates that it is feasible for ODS (or societies) to achieve a degree of distributional fairness due to sufficiently high mutual trust among the majority of agents. This applies even to ODS that do not have a trusted, central control, and include some agents that violate fair procedures. In the social sciences, this could be interpreted as a society that was not developed strong central institutions or authorities.

On the other hand, the Fairness Emergence hypothesis has an inherent practical value. First, if the FE hypothesis holds, then the problem of ensuring

fairness in an open, distributed system without centralized control may have found a practical solution: it would suffice to use a good trust management system in order to provide fairness. Second, if the FE hypothesis was not to be true in realistic conditions, then a reputation (or trust management) system would allow the existence of a degree of unfairness between similar agents. Such a situation would be highly undesirable from the point of view of users of trust management systems, leading to a disincentive of their usage.

We have shown that the Fairness Emergence hypothesis applies in realistic conditions: in the presence of adversaries and in an unstable state of the system, and also in an open system where the presence of sellers was controlled by a trace from a real Internet auction site. Yet, this work also shows that the FE hypothesis does not apply universally. In particular, fairness emergence does not occur (or is very weak) if very little negative feedback is received by the reputation system. The FE hypothesis does not hold if the users of a reputation system are not sufficiently sensitive to reputation or do not have enough choice of transaction partners with a good enough reputation (this implies that if unfair agents were to be a large fraction of the population, fairness could not emerge).

Fairness Emergence among buyers was not observed in the open system if the system was not warmed up. The reason for this is that in the open system, some sellers are present only for a few transactions. If the reputation system does not have sufficient information about these sellers, the buyers cannot determine whether they are fair or unfair. It would be possible to initialize the reputations of sellers with a small value, but that would effectively exclude them from the system since it would make it impossible for a new seller to earn a higher reputation. There exists a practical way out of this difficulty: the transactions of new agents could be insured, until their reputation reaches a sufficiently high value. There also exists a practical threat that can lead to a lack of sufficient information about agents: if agents who have a low reputation can assume a new identity (an approach known as whitewashing), then fairness emergence would not occur. This behavior can only be prevented by using stronger authentication of agents.

We have studied the sensitivity of fairness emergence to discrimination attacks. While fairness emergence can still be observed when sellers discriminate a minority of buyers, it is not statistically significant. In simulations where the discriminating agents formed a majority of the population, the FE hypothesis does not hold.

From these results we can draw the following conclusions:

1. trust management (reputation) systems can improve distributional fairness in ODS,
2. trust management systems should explicitly consider fairness in their evaluation (also in the evaluation of their correctness).

Further research is necessary to establish the sensitivity of the FE hypothesis to more sophisticated attacks on reputation systems. Furthermore, it would

be desirable to investigate the emergence of fairness in more general trust management systems, for example in systems that make use of risk in decision support. Another possibility would be the use of transaction insurance together with a reputation system. Last but not least, the use of reputation in practical fair procedures would require a redesign of these procedures – in the light of our results, this is a promising direction for future research.

Chapter 5
Conclusion

The problems of fair distribution, assuring fair behavior, and enabling agents to make decisions under uncertainty on the basis of trust, are among the most challenging in informatics today. A wide variety of applications require the solutions of these problems. Some of these applications have been presented in this book, along with some proposed solutions that have been grouped in two areas of Trust Management (TM) and Fairness Management (FM).

The difficulty of solving FM and TM problems lies first in the difficulty of defining the two concepts in a way that can be made operational and treated by informatics. This book has attempted to show that this difficulty can be overcome. A variety of definitions of fairness and trust exist and have been described in sections 2.2 and 2.3. However, we have shown that it is possible to choose general and broad definitions that can be operationalized and are in agreement with human preferences, as shown by a number of empirical studies that are described in this book. This means that not only are these precise definitions that can be treated mathematically, but also human users of Open Distributed Systems will likely approve of the behavior of TM and FM systems that are based on the proposed definitions of trust and fairness.

This book has attempted to show that not only can we define trust and fairness, but that it is possible to solve practical problems of TM and FM management. In order to achieve this goal, it is necessary to rely on the knowledge of the social sciences and on results of empirical observations and experiments. This knowledge is necessary to improve the organization of virtual ODS, to express important social concepts (such as trust, reputation, distrust) in computational systems, and also to design better algorithms that are used in ODS to realize social goals of fairness and trust. In this respect, this book attempted to provide a synthesis of the relevant knowledge in social sciences and the state of the art research in information science. The book gives several examples of computational expressions of social concepts, such as the operational definition of distributive fairness that is used to solve practical problems of grid scheduling or computer network dimensioning, or the use of computational trust in Internet auctions, recommendation systems

A. Wierzbicki: Trust & Fairness in Open, Distributed Sys., SCI 298, pp. 225–227.
springerlink.com

and P2P systems. These examples also present concrete algorithms, such as the CloseLook algorithm for trust propagation (see section 2.2.7) or the Equitable Walk algorithm for discovering equitable solutions in complex fair distribution problems (see section 4.1.1.4).

The two areas of Trust and Fairness Management are closely related. Even some basic definitions of fairness (as a justified expectation) and trust (as an expectation of behavior) allow us to conclude that an agent will be trusted if he behaves fairly (see section 2.2.1.1). These definitions and the reasoning based on them require the additional assumption of agent rationality and that the context of encounters includes rules of fair behavior that are known to all agents. While these assumptions do not hold in all situations for human agents, they form an interesting guideline for the design of TM and FM systems. Indeed, many such systems attempt to express rules of fair behavior, such as policies of access control or privacy policies. Another interesting example of fairness rules was described in section 3.5.2.2, where it was found that almost all nonpositive comments sent by users of Internet auctions to the TM system concern the violation of procedural fairness rules that form a code of conduct in Internet auctions. This finding gives further support to the hypothesis of a relationship between trust and fairness. A promising direction of future research would be the development of practical systems for the expression of rules of fair behavior that can be automatically be verified by TM and FM systems.

Other relationships between FM and TM exist. An example is the phenomenon of fairness emergence that has been studied in some detail in this book (see section 4.3). Fairness emergence is due to the existence and application of a TM system. It can be particularly useful in fully distributed settings, because Fairness Management often requires some form of centralized control, especially for providing distributive fairness. The ability of using a fully distributed TM system to enable the emergence of a certain degree of distributive fairness is therefore important. Further research could consider the question of whether TM systems can be used in other fair procedures, for example in sealed auctions, in order to remove the dependence of these procedures on central control by trusted agents. The validity of the Fairness Emergence hypothesis also has a theoretical significance. It demonstrates that in an ODS (or a society) that has little established central control, the use of trust can lead to the emergence of distributional fairness. Both concepts play an important part in social development, and since it is generally agreed in the social sciences that distributional fairness and equity play an important role in social welfare, it is important to understand that mutual trust of a majority of agents can cause their emergence.

Throughout this book, the question of distribution of control has been considered. There seems to exist a very general tradeoff between the distribution of control, security and trust. Usually, many fair procedures (or the implementation of solutions for fair distribution problems) require the existence of a trusted agent that has some measure of central control over the system.

The validity of the fairness emergence hypothesis seems to suggest that it would be possible to remove this reliance by introducing a TM system that explicitly deals with trust relationships. A straightforward approach would simply allow us to choose an arbitrary, trusted coordinator from among the most trusted agents in the system; this would at least remove the need of relying on a central controller that is trusted a priori without the possibility of verification. More complex approaches could modify the execution of fair procedures dynamically on the basis of changing trust relations.

The development of cloud computing and the increased availability of cheap computing resources has demonstrated that the future of Internet-based computation systems may lie in a hybrid architecture. It is no longer considered scalable to maintain expensive and resource-consuming centralized client-server applications; on the other hand, the pure distribution of the Peer-to-Peer model cannot guarantee sufficient reliability and security for many applications. The hybrid model would combine the best features of the two competing architectures, allowing systems to move seamlessly (and in a manner transparent to users) from a more centralized to a more distributed model. In order to make this vision possible, TM systems must become sufficiently reliable to allow the hybrid system to make critical decisions on the degree of trust in the available resources.

Another trend is the increasing control of Web users not just over the information provided through Web2.0 applications, but also over the functions of information processing themselves. With browser plugins or extensions, Javascript and Ajax, users can participate in the execution of information processing functions of the system, not just in the creation of content. In such new information systems, the questions of fairness and trust will become even more important.

As the e-society continues to develop, it can be expected that its virtual social functions will have an increased level of sophistication. More and more social processes will be taking place online. Internet-based social systems have the capacity of not just replacing, but broadening and deepening our real social relations. Coupled with an increasing pace of social and technological innovation that can create completely new social needs, this process will further increase the importance of fairness and trust in open distributed systems.

References

1. Aberer, K., Datta, A., Hauswirth, M.: A decentralised public key infrastructure for customer-to-customer e-commerce. International Journal of Business Process Integration and Management 1, 26–33 (2005)
2. Abrams, Z., McGrew, R., Plotkin, S.A.: A non-manipulable trust system based on eigentrust. SIGecom Exchanges 5(4), 21–30 (2005)
3. Abd-El-Malek, M., Granger, G.R., Goodson, G.R., Reiter, M.K., Wylie, J.J.: Fault-Scalable Byzantine Fault-Tolerant Services. In: Proc. SOSP 2005 (2005)
4. Agnetis, A., Mirchandani, P.B., Pacciarelli, D., Pacifici, A.: Scheduling Problems with Two Competing Agents. Operations Research 52(2), 229–242 (2004)
5. Angel, E., Bampis, E., Pascual, F.: The price of approximate stability for a scheduling game problem. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 157–166. Springer, Heidelberg (2006)
6. Aberer, K., Datta, A., Hauswirth, M.: A decentralised public key infrastructure for customer-to-customer e-commerce. International Journal of Business Process Integration and Management 1, 26–33 (2005)
7. Axelrod, R.: The Evolution of Cooperation. Basic Books, New York (1984)
8. Ba, S., Whinston, A.B., Zhang, H.: Building trust in online auction markets through an economic incentive mechanism. Decision Support Systems 35, 273–286 (2003)
9. Barber, B.: Logic and Limits of Trust. Rutgers University Press, New Jersey (1983)
10. Barr, A.: Trust and expected trustworthiness: An experimental investigation (2001)
11. Baughman, N.E., Levine, B.: Cheat-proof playout for centralized and distributed online games. In: INFOCOM 2001, pp. 104–113 (2001)
12. Berglund, E.J., Cheriton, D.R.: Amaze: A multiplayer computer game. IEEE Software 2(1), 30–39 (1985)
13. Bertsekas, D., Gallager, R.: Data Networks. Prentice-Hall, Englewood Cliffs (1987)
14. Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized trust management. In: Proc. of 1996 IEEE Symposium of Security and Privacy, Oakland, CA, May 1996, pp. 164–173 (1996)
15. Blazewicz, J.: Scheduling in Computer and Manufacturing Systems. Springer, Heidelberg (1996)

16. Bonald, T., Massoulie, L.: Impact of fairness on internet performance. In: Proceedings of ACM Sigmetrics, pp. 82–91 (2001)
17. Brams, S., Taylor, A.: Fair Division: From cake-cutting to dispute resolution. Cambridge University Press, Cambridge (1996)
18. Brams, S.: Designing better voting and fair-division procedures. Princeton University Press, Princeton (2008)
19. Buczynski, A., Przepiorkowski, A.: Demo: An Open Source Tool for Partial Parsing and Morphosyntactic Disambiguation. In: Proceedings of LREC (2008)
20. Buczynski, A., Wawer, A.: Shallow parsing in sentiment analysis of product reviews. In: Proceedings of the Partial Parsing workshop at LREC 2008 (2008)
21. Buetow, K.H.: Cyberinfrastructure: Empowering a "third way" in biomedical research. Science 308, 821–824 (2005)
22. Burmester, M., Desmedt, Y.G.: Is hierarchical public-key certification the next target for hackers? Communication of the ACM 47(8), 68–74 (2004)
23. Buyya, R., Abramson, D., Venugopal, S.: The grid economy. In: Special Issue on Grid Computing, vol. 93, pp. 698–714. IEEE Press, Los Alamitos (2005)
24. Calsamiglia, X., Kirman, A.: A unique informationally efficient and decentralized mechanism with fair outcomes. Econometrica 61(5), 1147–1172 (1993)
25. Camerer, C.F.: Behavioral Game Theory: Experiments in Strategic Interaction. Princeton University Press, Princeton (2003)
26. Carra, D., Neglia, G., Michiardi, P.: On the impact of greedy strategies in bittorrent networks: The case of bittyrant. In: Wehrle, K., Kellerer, W., Singhal, S.K., Steinmetz, R. (eds.) Peer-to-Peer Computing, pp. 311–320. IEEE Computer Society, Los Alamitos (2008)
27. Castelfranchi, C., Conte, R., Paolucci, M.: Normative reputation and the costs of compliance. J. Artificial Societies and Social Simulations 1(3) (1998)
28. Castro, M., Liskov, B.: Practical Byzantine Fault Tolerance. In: Proc. OSDI (1999)
29. Colombo, M., Martinelli, F., Mori, P., Petrocchi, M., Vaccarelli, A.: Fine grained access control with trust and reputation management for globus, pp. 1505–1515 (2007)
30. Cornelli, F., Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., Samarati, P.: Choosing reputable servents in a p2p network. In: WWW, pp. 376–386 (2002)
31. Covey Stephen, M.R.: The Speed of Trust, CoveyLink (2006)
32. Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., Samarati, P., Violante, F.: A reputation-based approach for choosing reliable resources in peer-to-peer networks. In: Atluri, V. (ed.) ACM Conference on Computer and Communications Security, pp. 207–216. ACM, New York (2002)
33. Damiani, E., De Capitani di Vimercati, S., Samarati, P., Viviani, M.: A WOWA-based Aggregation Technique on Trust Values Connected to Metadata (2006)
34. Dellarocas, C.: Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In: EC 2000: Proceedings of the 2nd ACM conference on Electronic commerce, pp. 150–157. ACM, New York (2000)
35. Denda, R., Banchs, A., Effelsberg, W.: The fairness challenge in computer networks. In: Crowcroft, J., Roberts, J., Smirnov, M.I. (eds.) QofIS 2000. LNCS, vol. 1922, pp. 208–220. Springer, Heidelberg (2000)

36. Deutsch, M.: Cooperation and Trust: Some Theoretical Notes. In: Nebraska Symposium on Motivation. Nebraska University Press (1962)
37. Deutsch, M.: The Resolution of Conflict. Yale University Press, New Haven (1973)
38. Deutsch, M.: Equity, equality, and need: What determines which value will be used as the basis of distributive justice? Journal of Social Issues 31, 137–149 (1975)
39. Deutsch, M.: Distributive justice: A social psychological perspective. Yale University Press, New Haven (1985)
40. Deutsch, M.: Experimental studies of the effects of different systems of distributive justice. In: Social comparison, social justice, and relative deprivation, pp. 151–164. Lawrence Erlbaum, Hillsdale (1987)
41. Dewan, S., Hsu, V.: Trust in electronic markets: Price discovery in generalist versus specialty online auctions (2001)
42. Douceur, J.: Peer-to-Peer Systems. The Sybil attack, pp. 251–260. Springer, Heidelberg (2002)
43. Dutot, P.F., Eyraud, L., Mounié, G., Trystram, D.: Scheduling on large scale distributed platforms: from models to implementations. Int. J. Found. Comput. Sci. 16(2), 217–237 (2005)
44. Elgesem, E.: Normative structures in trust management. In: Stølen, K., Winsborough, W.H., Martinelli, F., Massacci, F. (eds.) iTrust 2006. LNCS, vol. 3986, pp. 48–61. Springer, Heidelberg (2006)
45. Fehr, E., Schmidt, K.: Theories of fairness and reciprocity - evidence and economic applications. Working Paper, No. 75 (2001)
46. Feitelson, D.G., Rudolph, L., Schwiegelshohn, U.: Parallel job scheduling — A status report. In: Feitelson, D.G., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2004. LNCS, vol. 3277, pp. 1–16. Springer, Heidelberg (2005)
47. Feurbaey, M.: Fairness, Responsibility and Welfare. Oxford University Press, Oxford (2008)
48. Foster, I.: What is the grid
49. Foster, I.: Globus toolkit version 4: Software for service-oriented systems. In: Jin, H., Reed, D., Jiang, W. (eds.) NPC 2005. LNCS, vol. 3779, pp. 2–13. Springer, Heidelberg (2005)
50. Foster, I., Kesselman, C. (eds.): The Grid 2. Blueprint for a New Computing Infrastructure. Elsevier, Amsterdam (2004)
51. Friedman, J.: A non-cooperative equilibrium for supergames. Review of Economic Studies 38, 1–12 (1971)
52. Fudenberg, D., Tirole, J.: Game Theory. MIT Press, Cambridge (1991)
53. Gambetta, D.: Can We Trust Trust? In: Trust: Making and Braking of Cooperative Relations. Basil Blackwell, Oxford (1988)
54. Golbeck, J.: Computing and Applying Trust in Web-based Social Networks. PhD thesis, University of Maryland (2005)
55. Gregg, D.G., Scott, J.E.: A typology of complaints about eBay sellers. ACM 51, 69–74 (2008)
56. Gruber, R., Keller, V., Kuonen, P., Sawley, M.-C., Schaeli, B., Tolou, A., Torruella, M., Tran, T.-M.: Towards an intelligent grid scheduling system. In: Wyrzykowski, R., Dongarra, J., Meyer, N., Waśniewski, J. (eds.) PPAM 2005. LNCS, vol. 3911, pp. 751–757. Springer, Heidelberg (2006)
57. Guha, R., Kumar, R., Raghavan, P., Tomkins, A.: Propagation of trust and distrust. In: WWW 2004: Proceedings of the 13th international conference on World Wide Web, New York, NY, USA, pp. 403–412 (2004)

58. Hang, H.C.-W., Wang, Y.: Operators for propagating trust and their evaluation in social networks. In: Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems (2009)
59. Hardin, R.: Conceptions and Explanations of Trust. In: Trust in Society, vol. 2, pp. 3–39. Russel Sage Foundation, New York (2001)
60. Hazeyama, H., Iimura, T., Kadobayashi, Y.: Zoned federation of game servers: a peer-to-peer approach to scalable multi-player online games. In: Proc. ACM SIGCOMM, pp. 116–120 (2004)
61. Henrich, J., Boyd, R., Bowles, S., Camerer, C., Fehr, E., Gintis, H., McElreath, R.: In search of homo economicus: Behavioral experiments in 15 small-scale societies. Economics and Social Behavior 91(2), 73–78 (2002)
62. Houser, D., Wooders, J.: Reputation in internet auctions: Theory and evidence from ebay. Technical report, University of Arizona (2001)
63. Jaffe, J.: Bottleneck flow control. IEEE Transactions on Communications 7, 207–237 (1980)
64. Jelasity, M., Kermarrec, A.-M.: Ordered Slicing of Very Large-Scale Overlay Networks. In: P2P 2006: Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing, Washington, DC, USA, pp. 117–124. IEEE Computer Society, Los Alamitos (2006)
65. Josang, A., Ismail, R.: The Beta Reputation System. In: Proceedings of the 15th Bled Electronic Commerce Conference (2002), www.home/tomek/pliki/_dokumenty/trust_reputation_recomendation_itp/_BETA_REPUTATION_COMBINE_FEEDBACK_JI2002-Bled.pdf:PDF
66. Johnson-George, C., Swap, W.C.: Measurement of specific interpersonal trust: Construction and validation of a scale to assess trust in a specific other. Journal of Personality and Social Psychology 43(6), 1306–1317 (1982)
67. Josang, A., Keser, C., Dimitrakos, T.: Can we manage trust? In: Herrmann, P., Issarny, V., Shiu, S.C.K. (eds.) iTrust 2005. LNCS, vol. 3477, pp. 93–107. Springer, Heidelberg (2005)
68. Josang, A., Lo Presti, S.: Analysing the relationship between risk and trust (2004)
69. Kamvar, S.D., Schlosser, T.M., Garcia-Molina, H.: The eigentrust algorithm for reputation management in p2p networks. In: Proceedings of the Twelfth International World Wide Web Conference (2003)
70. Kandori, M.: Social norms and community enforcement. The Review of Economic Studies 59, 63–80 (1992)
71. Kandori, M.: Introduction to repeated games with private monitoring. Journal of Economic Theory 102, 1–15 (2002)
72. Kaszuba, T., Rzadca, K., Wierzbicki, A.: Discovering the most trusted agents without central control. In: Proceedings EUC 2008-Workshop accepted for inclusion (2008)
73. Kaszuba, T., Hupa, A., Wierzbicki, A.: Comment classification for Internet auction platforms. In: Grundspenkis, J., Kirikova, M. (eds.) ADBIS 2009 Workshops. LNCS, vol. 5968, pp. 129–136. Springer, Heidelberg (2009)
74. Kazemi, A.: Distributive Preferences in Social Dilemmas. Department of Psychology, Goteborg University, Sweden (2006)
75. Kelly, F., Mauloo, A., Tan, D.: Rate control for communication networks: shadow prices, proportional fairness and stability. Journal of the Operations Research Society 49, 206–217 (1997)

76. Kenyon, C., Cheliotis, G.: Grid resource commercialization: economic engineering and delivery scenarios. In: Nabrzyski, J., Schopf, J.M., Weglarz, J. (eds.) Grid resource management: state of the art and future trends, pp. 465–478. Kluwer Academic Publishers, Norwell (2004)
77. Kemeny, J., Snell, L.: Mathematical models in the social sciences. Ginn, New York (1962)
78. King-Casas, B., Tomlin, D., Anen, C., Camerer, C.F., Quartz, S.R., Read Montague, P.: Getting to know you: Reputation and trust in a two-person economic exchange. Science 308, 78–83 (2005)
79. Klein, R.S., Luss, H., Rothblum, U.G.: Minimax resource allocation problems with resource-substitutions represented by graphs. Operations Research 41, 959–971 (1993)
80. Xu, W., Knutsson, B., Lu, H., Hopkins, B.: Peer-to-peer support for massively multiplayer games. In: Proc. IEEE INFOCOM 2004, pp. 107–117 (2004)
81. Komorita, S.S., Parks, C.D.: Social Dilemmas. Brown and Benchmark, Madison (1994)
82. Komorita, S.S., Parks, C.D.: Interpersonal relations: Mixed-motive interaction. Annual Review of Psychology 46, 183–207 (1995)
83. Kostreva, M., Ogryczak, W.: Linear optimization with multiple equitable criteria. RAIRO Operations Research 33, 275–297 (1999)
84. Kotla, R., Alvisi, L., Dahlin, M., Clement, A., Wong, E.: Zyzzyva: Speculative Byzantine Fault Tolerance. In: Proc. SOSP 2007 (2007)
85. Kuter, U., Golbeck, J.: Sunny: A new algorithm for trust inference in social networks using probabilistic confidence models. In: AAAI, pp. 1377–1382 (2007)
86. Kuter, U., Nau, D., Gossing, D., Lemmer, J.F.: Interactive course-of-action planning using causal models. In: Proceedings of the Third International Conference on Knowledge Systems for Coalitionslition Operations KSCO 2004 (2004)
87. Kwok, Y.-K., Song, S., Hwang, K.: Selfish grid computing: Game-theoretic modeling and nas performance results. In: Proceedings of CCGrid (2005)
88. Lee, S., Sherwood, R., Bhattacharjee, B.: Cooperative peer groups in nice. In: Proc. INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 2, pp. 1272–1282. IEEE, Los Alamitos (2003)
89. Legout, A., Liogkas, N., Kohler, E., Zhang, L.: Clustering and sharing incentives in bittorrent systems. SIGMETRICS Perform. Eval. Rev. 35(1), 301–312 (2007)
90. Levien, R.: Attack Resistant Trust Metrics. PhD thesis, UC Berkeley (2003)
91. Levien, R., Aiken, A.: Attack-resistant trust metrics for public key certification. In: 7th USENIX Security Symposium, pp. 229–242 (1998)
92. Lewis, J.D., Weigert, A.: Trust as a social reality. Social Forces 63, 967–985 (1985)
93. Liang, Z., Shi, W.: Pet: A personalized trust model with reputation and risk evaluation for p2p resource sharing (2005)
94. Liogkas, N., Nelson, R., Kohler, E., Zhang, L.: Exploiting bittorrent for fun (but not profit) (2006)
95. Lissowski, G.: Zasady sprawiedliwego podziału dóbr (Principles of Fair Distribution of Goods). Wydawnictwo Naukowe Scholar (2008)

96. Lissowski, G., Ašwistak, P.: Choosing the best social order: New principles of justice and normative dimensions of choice. American Political Science Review 89, 74–96 (1995)
97. Liu, X., Datta, A., Rzadca, K., Lim, E.-P.: StereoTrust: a group based personalized trust model. In: CIKM 2009: Proceeding of the 18th ACM conference on Information and knowledge management, pp. 7–16. ACM, New York (2009), http://doi.acm.org/10.1145/1645953.1645958
98. Liu, J., Jin, X., Wang, Y.: Agent-based load balancing on homogeneous minigrids: Macroscopic modeling and characterization. IEEE Trans. on Parallel and Distributed Systems 16(7), 586–598 (2005)
99. Liu, L., Zhang, S., Dong Ryu, K., Dasgupta, P.: R-chain: A self-maintained reputation management system in p2p networks. In: Bader, D.A., Khokhar, A.A. (eds.) ISCA PDCS, pp. 131–136. ISCA (2004)
100. Locher, T., Moor, P., Schmid, S., Wattenhofer, R.: Free riding in bittorrent is cheap. In: In Proc. of HotNets-V (2006)
101. Luhmann, N.: Familiarity, confidence and trust: problems and alternatives. In: Trust: Making and Braking of Cooperative Relations. Basil Blackwell, Oxford (1988)
102. Luss, H.: On equitable resource allocation problems: a lexicographic minimax approach. Operations Research 47, 361–378 (1999)
103. Macy, M.W., Skvoretz, J.: The evolution of trust and cooperation between strangers: A computational model. American Sociological Review 63(5), 638–660 (1998)
104. Malaga, R.: Web-based reputation management systems: Problems and suggested solutions. Electronic Commerce Research, 1 (2001)
105. Marchal, L., Yang, Y., Casanova, H., Robert, Y.: A realistic network/application model for scheduling divisible loads on large-scale platforms. In: Proceedings of IPDPS, vol. 01, p. 48b (2005)
106. Marchi, E., Oviedo, J.A.: Lexicographic optimality in the multiple objective linear programming: the nucleolar solution. European Journal on Operations Research 57, 355–359 (1992)
107. Marsh, S.P., Dibben, M.R.: Trust, untrust, distrust and mistrust - an exploration of the dark(er) side. In: Herrmann, P., Issarny, V., Shiu, S.C.K. (eds.) iTrust 2005. LNCS, vol. 3477, pp. 17–33. Springer, Heidelberg (2005)
108. Marsh, S.P.: Formalising trust as a computational concept. PhD thesis, University of Stirling (April 1994)
109. Massa, P.: Downloaded epinions dataset (2003)
110. Martin, J.-P., Alvisi, L.: Fast Byzantine Consensus. In: Proc. IEEE TODSC 2006 (2006)
111. McAllister, D.J.: Affect- and cognition-based trust as foundations for interpersonal cooperation in organizations. Academy of Management Journal 38, 24–59 (1995)
112. Melnik, M.I., Alm, J.: Does a seller's ecommerce reputation matter? evidence from ebay auctions. The Journal of Industrial Economics L(3) (September 2002)
113. Meulpolder, M., Pouwelse, J.A., Epema, D.H.J., Sips, H.J.: Bartercast: Fully distributed sharing-ratio enforcement in bittorrent (2008)
114. Mo, J., Walrand, J.: Fair end-to-end window-based congestion control. IEEE/ACM Transactions on Networking 8, 556–567 (2000)

115. Mohtashemi, M., Mui, L.: Evolution of indirect reciprocity by social information: the role of trust and reputation in evolution of altruism. Journal of Theoretical Biology 223, 523–531 (2003)
116. Mol, J.J.-D., Pouwelse, J.A., Epema, D.H.J., Sips, H.J.: Free-riding, fairness, and firewalls in p2p file-sharing. In: Wehrle, K., Kellerer, W., Singhal, S.K., Steinmetz, R. (eds.) Peer-to-Peer Computing, pp. 301–310. IEEE Computer Society, Los Alamitos (2008)
117. Molm, L., Peterson, G., Takahashi, N.: Power in negotiated and reciprocal exchange. American Sociological Review 64, 876–890 (1999)
118. Mui, L.: Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks. PhD thesis, Massachusets Institute of Technology (December 2002)
119. Nee, V., Sanders, J.: Trust in Ethnic Ties: Social Capital and Immigrants. In: Trust in Society, vol. 2, pp. 374–392. Russel Sage Foundation, New York (2001)
120. Ogryczak, W.: Linear and Discrete Optimization with Multiple Criteria: Preference Models and Applications to Decision Support (in Polish). Warsaw University Press, Warsaw (1997)
121. Ogryczak, W., Sliwinski, T.: On Decision Support Under Risk by the WOWA Optimization. In: Mellouli, K. (ed.) ECSQARU 2007. LNCS (LNAI), vol. 4724, pp. 779–790. Springer, Heidelberg (2007)
122. Ogryczak, W.: On principles of fair resource allocation for importance weighted agents. In: Proc. First International Conference on Social Informatics (2009)
123. Ogryczak, W., Pi/oro, M., Tomaszewski, A.: Telecommunication network design and max-min optimization problem. Journal of Telecommunication and Information Technology 3, 43–56 (2005)
124. Ogryczak, W., Sliwinski, T.: On equitable approaches to resource allocation problems: the conditional minimax solution. Journal of Telecommunication and Information Technology 3, 40–48 (2002)
125. Ogryczak, W., Śliwiński, T., Wierzbicki, A.: Fair resource allocation schemes and network dimensioning problems. Journal of Telecommunication and Information Technology 3, 34–42 (2003)
126. Ogryczak, W., Tamir, A.: Minimizing the sum of the k largest functions in linear time. Information Processing Letters 85, 117–122 (2003)
127. Ogryczak, W., Wierzbicki, A.: On multi-criteria approaches to bandwidth allocation. Control and Cybernetics 33, 427–448 (2004)
128. Ogryczak, W., Kostreva, M.M., Wierzbicki, A.: Equitable aggregations and multiple criteria analysis. European Journal of Operational Research 158, 362–377 (2004)
129. Ogryczak, W., Wierzbicki, A., Milewski, M.: A Multi-Criteria Approach to Fair and Efficient Bandwidth Allocation. OMEGA 36, 451–463 (2008)
130. Okuno-Fujiwara, M., Postlewaite, A.: Social norms and random matching games. Games and Economic Behavior 9, 79–109 (1995)
131. Piatek, M., Isdal, T., Anderson, T., Krishnamurthy, A., Venkataramani, A.: Do incentives build robustness in bittorrent (awarded best student paper). In: NSDI. USENIX (2007)
132. Philip, J.: The General Inquirer: A Computer Approach to Content Analysis. The MIT Press, Cambridge (1966)

133. Pióro, M., Medhi, D.: Routing, Flow and Capacity Design in Communication and Computer Networks. Morgan Kaufmann Publishing, San Francisco (2004)
134. Pollock, G.B., Dugatkin, L.A.: Reciprocity and the evolution of reputation. Journal of Theoretical Biology 159, 25–37 (1992)
135. Pouwelse, J.A., Garbacki, P., Wang, J., Bakker, A., Yang, J., Iosup, A., Epma, D.H.J., Reinders, M., Steen, M.V., Sips, H.J.: Tribler: A social-based peer-to-peer system. In: The 5th International Workshop on Peer-to-Peer Systems (IPTPS 2006), pp. 1–6 (2006)
136. Putnam, R.: Bowling Alone: The Collapse and Revival of American Community. Simon and Schuster, New York (2000)
137. Rawls, J.: The Theory of Justice. Harvard University Press (1971)
138. Repast Organization for Architecture and Development (2003), `http://repast.sourceforge.net`
139. Resnick, P., Kuwabara, K., Zeckhauser, R., Friedman, E.: Reputation systems. Communications of the ACM 43(12), 45–48 (2000)
140. Resnik, P., Zeckhauser, R.: Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system. Advances in Applied Microeconomics 11 (2002)
141. Resnik, P., Zeckhauser, R.: Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system. Advances in Applied Microeconomics 11 (2002)
142. Resnik, P., Zeckhauser, R., Swanson, J., Lockwood, K.: The value of reputation on ebay: a controlled experiment. Technical report, School of Information, University of Michigan (2004)
143. Ruohomaa, S., Kutvonen, L.: Trust management survey. In: Herrmann, P., Issarny, V., Shiu, S.C.K. (eds.) iTrust 2005. LNCS, vol. 3477, pp. 77–92. Springer, Heidelberg (2005)
144. Ruohomaa, S., Kutvonen, L., Koutrouli, E.: Reputation management survey. In: ARES, pp. 103–111. IEEE Computer Society, Los Alamitos (2007)
145. Rzadca, K., Trystram, D., Wierzbicki, A.: Fair game-theoretic resource management in dedicated grids. In: Proc. of the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007), pp. 343–350. IEEE Computer Society, Los Alamitos (2007)
146. Sabater-Mir, J., Paolucci, M.: On representation and aggregation of social evaluations incomputational trust and reputation models. International Journal of Approximate Reasoning 46, 458–483 (2007)
147. Selberg, E.W.: How to stop a cheater: Secret sharing with dishonest participation. Technical report, Carnegie Mellon University (1993)
148. Selcuk, A.A., Uzun, E., Pariente, M.R.: A reputation-based trust management system for p2p networks. In: CCGRID 2004: Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid, Washington, DC, USA, pp. 251–258. IEEE, Los Alamitos (2004)
149. Sen, A.: Collective Choice and Social Welfare. Holden Day (1970)
150. Shorrocks, A.: Ranking income distributions. Economica 50(197), 3–17 (1983)
151. Singh, A., Liu, L.: Trustme: Anonymous management of trust relationships in decentralized p2p systems. In: Shahmehri, N., Graham, R.L., Caronni, G. (eds.) Peer-to-Peer Computing, pp. 142–149. IEEE Computer Society, Los Alamitos (2003)
152. Song, S., Hwang, K., Zhou, R., Kwok, Y.-K.: Trusted p2p transactions with fuzzy reputation aggregation. IEEE Internet Computing 9(6), 24–34 (2005)

153. Song, Y.J., van Renesse, R.: Bosco: One-Step Byzantine Asynchronous Consensus. In: Taubenfeld, G. (ed.) DISC 2008. LNCS, vol. 5218, pp. 438–450. Springer, Heidelberg (2008)
154. IETF, SPKI Working group, `http://www.ietf.org`
155. Srivatsa, M., Xiong, L., Liu, L.: Trustguard: countering vulnerabilities in reputation management for decentralized overlay networks. In: WWW 2005: Proceedings of the 14th international conference on World Wide Web, pp. 422–431. ACM, New York (2005)
156. Stolle, D.: Clubs and Congregations: The Benefits of Joining an Association. In: Trust in Society, vol. 2. Russel Sage Foundation, New York (2001)
157. Sztompka, P.: Trust: A Sociological Theory. Cambridge University Press, Cambridge (1999)
158. Sztompka, P.: Zaufanie. Fundament Spo?eczen'stwa (Trust. A Foundation of Society). Wydawnictwo Znak (2007)
159. Tang, A., Wang, J., Low, S.H.: Is fair allocation always inefficient. In: IEEE INFOCOM (2004)
160. Terazona: Zona application framework white paper (2002)
161. Thomson, W.: Fair allocation rules. Working Paper 539, University of Rochester (2007)
162. Thibaut, J., Walker, L.: Procedural justice: A psychological analysis. Lawrence Erlbaum, Hillsdale (1975)
163. Tompa, M., Woll, H.: How to share a secret with cheaters. Technical Report Research Report RC 11840, IBM Research Division (1986)
164. Torra, V.: The weighted OWA operator. International Journal of Intelligent Systems 12, 153–166 (1997)
165. Trivers, R.: Social Evolution. Benjamin Cummings, Menlo Park (1985)
166. Tyler, T.R., Smith, H.J.: The handbook of social psychology. In: Social justice and social movements, vol. 2, pp. 595–629. McGraw-Hill, Boston (1998)
167. Vanstone, S.A., Menezes, J., van Oorschot, P.C.: Handbook of applied cryptography. CRC Press, Boca Raton (1996)
168. Volper, D.E., Oh, J.C., Jung, M.: Game theoretical middleware for cpu sharing in untrusted p2p environment. In: Proceedings of PDCS (2004)
169. Varandharajan, V., Zhao, W., Mu, J.: A secure mental poker protocol over the internet. In: Proc. Australasian Information Security Workshop (AISW2003), pp. 105–109 (2003)
170. Wang, Y., Vassileva, J.: Trust and reputation model in peer-to-peer networks. In: P2P 2003: Proceedings of the 3rd International Conference on Peer-to-Peer Computing, Washington, USA, p. 150. IEEE, Los Alamitos (2003)
171. Welch, M.R., Rivera, R.E.N., Conway, B.P., Yonkoski, J., Lupton, P.M., Giancola, R.: Determinants and consequences of social trust. Sociological Inquiry 75(4), 453–473 (2005)
172. Wierzbicki, A., Kaszuba, T.: Practical trust management without reputation in peer-to-peer games. In: Multiagent and Grid System, vol. 3(4), pp. 1–18. IOS Press, Amsterdam (2007); Guest Editors Prof. Pilar Herrero and Prof. Maria S. Perez and Editor-in-Chief Prof. R. Unland
173. Wierzbicki, A., Kaszuba, T., Nielek, R., Datta, A.: Trust and Fairness Management in P2P and Grid systems. In: Handbook of Research on P2p and Grid Systems for Service-oriented Computing: Models, Methodologies and Applications. IGI-Global (2009)

174. Wierzbicki, A.P.: A mathematical basis for satisficing decision making. Mathematical Modelling 3, 391–405 (1984)
175. Wierzbicki, A.P., Makowski, M., Wessels, J.: Model Based Decision Support Methodology with Environmental Applications. Kluwer, Dordrecht (2000)
176. Wierzbicki, A., Kucharski, T.: P2p scrabble. can p2p games commence? In: Proc. Fourth International IEEE Conference on Peer-to-Peer Computing, pp. 100–107. IEEE, Los Alamitos (2004)
177. Wierzbicki, A.: Trust enforcement in peer-to-peer massive multi-player online games. In: Meersman, R., Tari, Z. (eds.) GADA 2006. LNCS, vol. 4276, pp. 1163–1180. Springer, Heidelberg (2006)
178. Wierzbicki, A.: The case for fairness of trust management. Electr. Notes Theor. Comput. Sci. 197(2), 73–89 (2008)
179. Wierzbicki, A., Morzy, M.: The sound of silence: Mining implicit feedbacks to compute reputation. In: Spirakis, P.G., Mavronicolas, M., Kontogiannis, S.C. (eds.) WINE 2006. LNCS, vol. 4286, pp. 365–376. Springer, Heidelberg (2006)
180. Wierzbicki, A.P.: The Problem of Objective Ranking: Foundations, Approaches and Applications. Journal of Telecommunications and Information Technology 3, 15–23 (2008)
181. Wierzbicki, A., Wierzowiecki, G.: CloseLook: An Efficient and Correct Trust Propagation Algorithm. In: Joint work with my student Grzegorz Wierzowiecki (submitted to TrustBus 2010)
182. Wilson, E.O.: Sociobiology. Harvard University Press, Cambridge (1975)
183. Wolski, R., Brevik, J., Plank, J.S., Bryan, T.: Grid resource allocation and control using computational economies. In: Berman, F., Fox, G., Hey, A. (eds.) Grid Computing: Making The Global Infrastructure a Reality. John Wiley & Sons, Chichester (2003)
184. Xiong, L., Liu, L.: Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. IEEE Transactions on Knowledge and Data Engineering 16(7), 843–857 (2004)
185. Yager, R.R.: On ordered weighted averaging aggregation operators in multicriteria decision making. IEEE Transactions on Systems, Man and Cybernetics 18, 183–190 (1988)
186. Yager, R.: On the determination of strength of belief for decision support under uncertainty - Part I: generating strengths of belief. Fuzzy Sets and Systems 142, 117–128 (2004)
187. Yager, R.: On the determination of strength of belief for decision support under uncertainty - Part II: fusing strengths of belief. Fuzzy Sets and Systems 142, 129–142 (2004)
188. Yamagashi, T.: Trust in Society. In: Trust as a Form of Social Intelligence, vol. 2, pp. 121–147. Russel Sage Foundation, New York (2001)
189. Yoshino, H.: Byzantine Agreement Protocol Using Hierarchical Groups. In: Proc. International Conference on Parallel and Distributed Systems, pp. 64–70 (2005)
190. Young, H.P.: Equity: In Theory and Practice. Princeton University Press, Princeton (1994)
191. Yu, B., Singh, M.P., Sycara, K.: Developing trust in large-scale peer-to-peer systems. In: Proceedings of First IEEE Symposium on Multi-Agent Security and Survivability, pp. 1–10 (2004)
192. Zack, P., Knack, S.: Trust and Growth. Economic Journal (2001)

193. Zhao, H., Li, X.: H-trust: A robust and lightweight group reputation system for peer-to-peer desktop grid. In: Proc. 28th International Conference on Distributed Computing Systems Workshops ICDCS 2008, pp. 235–240, 17–20 (2008)
194. Zhou, R., Hwang, K.: Gossip-based reputation aggregation for unstructured peer-to-peer networks. Ipdps 0, 95 (2007)
195. Zhou, R., Hwang, K.: Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing. IEEE Trans. Parallel Distrib. Syst. 18(4), 460–473 (2007)
196. Ziegler, C.N., Lausen, G.: Propagation models for trust and distrust in social networks. Information Systems Frontiers 7(4-5), 337–358 (2005)

Index