Techniques for Peer Enforcement in Multiagent Networks

A dissertation submitted by
Adrián Perreau de Pinninck Bas
at Universitat Autònoma de Barcelona to
fulfill the degree of **PhD** in Computer Science.

Bellaterra, March 12, 2010

Director: **Carles Sierra Garcia**
Institut d'Investigació en Intel·ligència Artificial
Consell Superior d'Investigacions Científiques

Co-director: **Marco Schorlemmer**
Institut d'Investigació en Intel·ligència Artificial
Consell Superior d'Investigacions Científiques

a l'Avi

Contents

List of Figures

Abstract

Software systems are achieving a high level of complexity, thus becoming increasingly hard to manage through a centralised architecture. This is why decentralised software architectures are blooming, *e.g.,* multiagent systems, peer-to-peer networks, or sensor networks. Managing decentralised systems is complicated. Therefore, many decentralised systems maintain certain functions centralised, such as security. A dictionary definition of security is the degree of protection against danger, loss, and criminals. We take an alternative definition: the degree of satisfaction in interactions with others. Achieving security in a decentralised manner requires a different set of techniques than those used in centralised approaches. In this thesis we study enforcement techniques to be applied in a fully decentralised manner. In some cases, decentralised techniques are just generalisations of centralised techniques. Nonetheless, some decentralised techniques are unique to the mechanism used for the interaction process.

By modelling the software systems as multiagent networks, where each agent is connected to those agents it knows, and by defining simple interaction protocols, we have developed new enforcement techniques that can be applied by any agent in the system. The aim of these peer enforcement techniques is to reduce the sanctioned agent's ability to interact, bringing it one step closer to total ostracism. We have also developed sophisticated reputation modelling techniques that are robust against most widespread malicious attacks in order to help enforcing agents decide when to apply the enforcement techniques.

These peer enforcement techniques and reputation mechanisms have been evaluated analytically and experimentally in scenarios ranging from those that are closed and with a shared description of appropriate behaviour, to those that are open and with a subjective description of appropriate behaviour. The analytical results provide information about the limits of these techniques. Whereas the experimental results verify that applying the enforcement techniques has a positive effect in the average satisfaction experienced by the agents in the multiagent network. Furthermore, the experimental results have evaluated the extent of that positive effect and the types of scenarios for which they work best.

Acknowledgements

First of all I thank my parents. for without their support I would have never been able to pursue my dreams.

Furthermore, I thank Maria Salamó for getting me to do research, first by being my master thesis supervisor and later by forwarding me to the IIIA, where I did my PhD under the supervision of Carles Sierra, who always had his door open to hear my doubts out, and Marco Schorlemmer, who painstakingly taught me the art of mathematical modelling.

I also thank Gal Kaminka, Gery Gutnik, Stephen Cranefield, and Sharmila and Tony Savarimuthu for being such great hosts in my stays abroad. Not only did I get to do great research with them, but they also allowed me to take a glimpse into their culture.

Last but not least I am very grateful for having met Mariusz Nowostaski and Dorian Gaertner, with whom I have shared much more than I ever thought possible, and Manu Atencia, which I have enjoyed annoying and being annoyed by during these more than four wonderful years.

Chapter 1

Introduction

The field of Distributed Artificial Intelligence (DAI) studies how different processes can work together in order to solve specific goals. Multiagent Sytems (MAS) are a subset of DAI systems in which the different processes (a.k.a. agents) have been developed by different entities. Therefore, each agent may have its own goals which it wants to satisfy by interacting with other agents in the MAS.

An agent chooses to interact with other agents because it is not able to achieve its goals on its own. Nonetheless, interaction with other agents does not guarantee that it will achieve its goals. The agent must find those agents that can help it achieve its goals and interact with them. In an interaction each agent has some expectations as to how the other agents will behave. These expectations can be exchanged and agreed to before the interaction takes place (*e.g.,* as in signing a contract), they can be fixed as part of the environment in which the interaction takes place (*i.e.,* by having the system designer specify the norms that will govern interactions), or none of the above, thus having agents interact blindly with one another allowing expected behaviours to be learned through time. Having agents in an interaction know one anothers expectations does not guarantee that they will be satisfied with the interaction results. Agents need mechanisms to enforce the behaviours they expect from others.

Humans have had to deal with these same issues, thus it is an interesting exercise to see what solutions have been proposed for humans. According to Taylor [Taylor, 1982], enforcement of expected behaviours has been achieved in primitive human societies through techniques that can be categorised through one or more of the following:

- Persuasion - where an agent modifies the beliefs of other agents through reasoning, so that they will believe that following the expected behaviour is preferable (*e.g.,* John persuades Peter to drive on the right hand side of the road by explaining why doing so avoids collisions).

- Authority - where an agent can modify the beliefs of other agents through its endorsement of the expected behaviour without giving reasons about

why the behaviour is best (*e.g.,* Peter drives on the right hand side of the road because John, which is a government official, told him it is best).

- Power - where an agent can execute actions that change the probabilities of other agents achieving their goals (*e.g.,* Peter drives on the right hand side of the road to avoid being fined by John, which is a police officer).

- Physical Constraint - where an agent can bring about actions that impede other agents from continuing to interact (*e.g.,* Peter drives on the right hand side of the highway because it is impossible for him to drive on the left hand side, since John, the engineer in charge of building the highway, placed a barrier dividing the two sides).

Techniques that classify under persuasion involve a high degree of cognitive capabilities. Those classifying as authoritative either expect agents to have capabilities through which they can model each others degree of authority, or they must be hardcoded into their instincts. On the other hand, techniques based on power and physical constraint involve dependencies amongst agents which can be used as rewards or sanctions.

When human societies are looked upon for examples of enforcement techniques based on physical constraint, one starts by identifying those physical characteristics common to humans that can be used to sanction and reward them. These common characteristics are the fact that humans feel emotions, such as pain, pleasure, shame, and loneliness. All these emotions can be used in order to get a human to act in a certain way. For example, the threat of inflicting pain has been a common enforcement technique in many human societies. Nonetheless, as of now, most of these emotions are not present in artificial agents. One could make an exception by stretching the meaning of loneliness by matching it to a sense of gregariousness. In a way, artificial agents want to be in "company" of other agents, since they need them in order to achieve their goals. Nonetheless, the enforcement technique that uses the need to interact with other agents can also be classified under the power category.

In order to identify power-based enforcement techniques, one has to identify the resources that the agents need in order to achieve their goals. Using human societies again as an example, we observe that their basic needs to achieve goals are: energy, time, physical resources, and information. These can all be transferred in some way or other from one human to another. A human's energy and time can be used to help another human achieve its goal, *e.g.,* Tom can use his strength and time to build Anne's new cupboard. Physical resources are limited and access to them can be granted to other humans, *e.g.,* Britney can lend her car to Alan so that he can get to work. Finally, information can be spread to others, *e.g.,* Cecile can advise Diego on which are the best weather conditions and routes to reach the Aconcagua summit. The threat of resource access denial is a power-based enforcement technique.

In order to find techniques that work for artificial software agents, one must start by identifying the resources they need to achieve goals. Surprisingly, these are the same as those seen for humans: energy, time, physical resources, and

information. An agent needs energy as a power source in order to run in a computer. It also needs time for its computations, and information as an input for them. Finally, it also needs physical resources: CPU, memory, and bandwidth. Out of these physical resources it is only bandwidth that can be transferred. Unless we are dealing with mobile agents, which we are not in this thesis, .

The work in this thesis describes power-based enforcement techniques where bandwidth is the physical resource used as the incentive to achieve expected behaviour. By blocking access to the resources needed in order to interact, an agent can enforce behaviours on others. This thesis shows how blockage from bandwidth usage through a network of agents impacts the ability to interact of agents that do not exhibit the expected behaviours. Models for MAS structured as networks are described where these blocking techniques can be applied, and the impact of such enforcement techniques are shown analytically and experimentally. Structuring a MAS as a network is a natural phenomenon since the advent of the internet, which is a network of networks through which humans communicate. It is becoming a widespread occurrence ever since the appearance of social networks. Although these are virtual networks in a centralised application scenario.

1.1 Motivation

Many distributed systems have appeared since the internet became a well-known technology. Through these systems many users around the globe come together to interact with each other and achieve certain goals. At first these systems where closed in many ways. Access was limited, and so were the actions available to users. Under these conditions is was fairly easy to get users to exhibit the expected behaviour. This was achieved in a centralised manner by the system designers which enforced the behaviours they wanted.

As the users grew in numbers and the technology evolved to allow more personalisation, it became harder for the system designers to enforce the behaviours that would suit all users. In order to satisfy users better, distributed system designers would have to give the users enforcement capabilities of their own and allow expected behaviour to emerge with time other than engineer it beforehand.

In today's systems, the only way users can enforce their expected behaviours is by deciding not to interact with those that they believe will not satisfy their expectations. Either because they have not done so in the past, or because they have come to know about previous interactions and realised that they have incompatible expectations.

Another technique for enforcement is to get others not to interact with a specific agent. This is done indirectly in current systems by publishing interaction feedback so that others with similar expectations choose not to interact with the given agent. Current online systems have incorporated technologies that aid in this gossip gathering process in order to assess the probability that an agent will satisfy specific expectations. These technologies are known as Trust

Managements Frameworks (TMF). Nonetheless, TMFs do not empower agents with new enforcement methods per se. They just give agents better information tools so that they can decide when to interact with other agents.

The motivation in this thesis is to find new methods through which the probability of having a satisfactory interaction is increased. This is achieved by enforcing the expected behaviours. Particularly, we are interested in those enforcement techniques that are totally distributed, since centralised techniques would not be as robust and may suffer bottleneck problems. By distributed techniques we mean those that can be aplied by all agents in the system while still allowing agents to have personal expectations and to have their own policies as to how strictly they want to enforce. Furthermore, these policies must take into account that other agents will try to avoid the sanctions being applied. Therefore, it is very important that the techniques provided are robust to these evasion techniques.

A system in which these distributed enforcement techniques are available should allow different communities to emerge in an efficient manner. Each of these communities would be formed by a set of agents whose expectations on the behaviours of others are compatible. Something like this is already happening in the Internet, where communities with different interests have formed. The difference being that, when enforcement techniques are provided through technology, they are either not efficient or only available to a selected few.

1.2 Contribution

The main idea underlying all our contributions is to structure a multiagent system through a social network. In small human communities, such as indigenous tribes or villages, all members know each other personally and know what to expect from each other from the outcome of many previous interactions. As communities grow in size, it is hard for those that conform them to know each other as is the case of cities or groups of villages scattered through an area. In such cases the lack of information about others can be overcome through third parties. By depending on these third parties for the information, they are being given power which can be used as an enforcement technique.

Let us illustrate this through an example. Albert, which has lived in the small town of Aberdale all his life working at his uncle Bill's farm, wants to move to the neighbouring town of Springfield where he plans to work at another farm. Albert knows no one at Springfield, and none of the farmers at Springfield have any knowledge about Albert's capabilities at farm work. Notwithstanding, Albert has a recommendation letter from his uncle Bill, which is well known by Charlie (a Springfield farmer) from previous deals at different farmer markets. Since Bill vouches for Albert, Charlie is willing to employ Albert at his farm. The interaction in this example took place because of Bill's vouching through the recommendation letter. Had Albert done something in the past that did not satisfy Bill, he would have been able to deny Albert a recommendation letter, thus lowering Albert's possibilities of working at Springfield.

Our contribution consists of two protocols for interaction bootstrap that force agents to depend on their contacts in the social network, thus giving agents a degree of power over their contacts which can be used through different enforcement techniques we have proposed. We also provide a mathematical model for multiagent systems structured as social networks, and we give some analytical results showing that a decrease in unsatisfactory interactions can be achieved under certain conditions. Nonetheless, the model provided allows agents many degrees of freedom in their behaviour. In the analytical exploration, we narrowed this freedom through strict assumptions on their behaviour and their expectations. In order to make up for this, we have also realised experiments in more complex scenarios.

Through the experiments, we have tested to what extent the proposed enforcement techniques can increase the ratio of satisfactory interactions. Furthermore we also tested how well specific approaches fared against different enforcement evasion techniques that are known to be used by malicious agents in current systems.

1.3 Thesis Structure

Chapter 2 surveys the current state of the art. In that chapter the relationship to other research fields is studied, and so are recent developments in the field of multiagent systems, specifically those that deal with trust and reputation management and those that deal with group formation. Furthermore, other distributed systems where the proposed enforcement techniques can be used are also reviewed.

Then Chapter 3 defines the experimental methodology that has been followed throughout the experiments. The methodology has guided the experiment design, the data gathering, and the subsequent statistical analysis of the experimental data results from which we have validated the original hypothesis. A chapter has been dedicated to explaining the methodology since it has been used for the experiments in Chapters 4 and 5.

Chapters 4 and 5 are the core part of the thesis. These chapters provide the interaction bootstrap protocols through which agents can enforce the expected behaviours on others. Chapter 4 describes a protocol for interaction bootstrap in which an agent searches for an interaction partner which is *not known* before the protocol starts. This chapter provides the first set of enforcement techniques which can be embedded into the interaction bootstrap protocols. The definition of a satisfactory interaction in this chapter is engineered through norms and shared by all. Therefore, all agents have the same definition of a satisfactory interaction which is objectively verifiable by all. Furthermore, analytical results are provided which give an upper bound to the number of unsatisfactory (*i.e.,* norm violating) interactions when specific conditions are met in the multiagent network. Finally, the results of experiments are provided which show the outcomes in less restrictive scenarios.

In Chapter 5 a second interaction bootstrap protocol is provided. In this

case the partner with which the agent wants to interact is *known* from the beginning of the protocol. This difference allows for a new enforcement technique which can be added to those provided in Chapter 4. Nonetheless, the definition of a satisfactory interaction in this chapter is made subjective, *i.e.,* each agent has its own definition and these definitions are not necessarily known by other agents. Therefore, the analytical results in Chapter 4 no longer hold and a new analysis has been realised. Finally, the updated set of enforcement techniques are tested against other enforcement mechanisms through experiments. These experiments test the reduction of unsatisfactory interactions, and also test whether the enforcement techniques are robust against adversarial behaviours by malicious agents.

Finally, Chapter 6 wraps up the thesis by providing the limitations of the approach, and how they may be tackled. It also provides some examples as to how the enforcement techniques can be applied to currently functioning systems.

Chapter 2

State of the Art

This chapter provides the reader with insight into how this work relates to other works. Each section deals with a research area to which this thesis is related, either because the work in this thesis is part of the research area or because the works in that research area aim to solve the same problems as the ones in this thesis. Section 2.1 describes the different distributed application paradigms. In Section 2.2 the new wave of internet applications that use social network data is presented. Section 2.3 explains the different uses of norms to regiment a MAS, which is closely related to the work in Chapter 4. Section 2.4 gives an overview of the research into enforcement techniques in electronic scenarios. Section 2.5 deals with research into how the system can detect undesired behaviour. Section 2.6 shows research in game theory that looks for ways in which to make expected behaviour rational. Section 2.7 describes the main approaches to manage trust and reputation, and Section 2.8 describes the main approaches to managing electronic currencies. Both currencies and reputation can be used as a virtual resource to be used in power-based enforcement.

2.1 Distributed Systems

User satisfaction is a complex issue addressed by distributed systems. Distributed computing has been booming in different ways. The approaches taken and the metaphores vary depending on the intended use of the system. The main distributed computing approaches are: Multiagent Systems (MAS), Peer-to-Peer (P2P) networks, Mobile Ad-hoc Networks (MANET), Grid Computing, Service Oriented Architectures (SOA), and Sensor Networks.

MAS [Wooldridge, 2002] is a subfield of distributed artificial intelligence in which autonomous software agents interact with one another in order to achieve personal and global goals. These agents will interact with other agents with the expectation that the interaction will get them closer to achieving their personal goals. Sometimes this might mean that an agent will act in a way that the interaction is satisfactory to it, while at the same time it is unsatisfactory to the

other party. Depending on the MAS infrastructure interaction among agents in a MAS may or may not be mediated by other agents. Enforcement mechanisms can be used to reduce the amount of times this sort of activity happens. The goal of such mechanisms is to reduce the number of interactions where any of the parties was not satisfied with the outcome.

Peer-to-Peer (P2P) computer networks are a way to organize a group of computers so that certain resources may be shared. P2P networks have made it into the mainstream through the different file sharing applications that have been developed with this technology (*e.g.,* Naptser, BitTorrent, Gnutella). Nonetheless, there are many other applications developed using this technology, such as voice communications (*e.g.,* Skype), conversation technologies (*e.g.,* Jabber), anonymous content storage (*e.g.,* Freenet, Free Haven, Publius) and anti-censorship news accessing (*e.g.,* Red Rover). P2P networks are meant to be large scale. This makes it impossible for any single agent to have all the information regarding the rest of the network. Therefore, P2P networks provide mechanisms to search for this information, normally by routing requests through the network. The main design issues for P2P networks are robustness, and efficiency in finding network information. Unfortunately it is hard to achieve both. P2P systems deal with different aspects of robustness. Firstly, it is robustness against turnover, meaning that computers may join and leave whenever they want and this should not disturb the way the system works. Secondly, they look into robustness against free riding, where users that use resources do not share theirs in return. When free riders are too many, not enough resources may be shared to act as an incentive for sharing peers to stay in the network. Finally, since P2P systems tend to be open to all, some users may act maliciously. Such malicious activity tends to be non-satisfactory to other users. In all these cases enforcement mechanisms can be used to increase robustness.

Mobile Ad-hoc Networks (MANET) are a specific type of network devised to share bandwidth in order for different mobile nodes with wireless connectivity to communicate with one another. If two nodes are out of reach of one another through their wireless antenas, then they can use other nodes in reach as relays/routers so that communication channels can be established between them. The fact that nodes are mobile adds another challenge to the whole routing process. Furthermore, since mobile hardware has a limited power source the nodes in a MANET may be tempted to free ride by not routing other nodes' communications. It is important in this context to have an enforcement mechanism through which free riding is minimized. Otherwise, since most communications are mediated by other nodes, free-riding may break the connections.

Grid Computing tries to solve a different problem. A Grid is in effect a cluster of computers. The difference with standard cluster computer technology is that a Grid is heterogeneous, loosely coupled, and geographycally dispersed. The structure of a grid should allow different computers accross different parts of the world to share their computational power to solve certain problems. These problems or tasks are decomposed and sent to other computers in the grid. When each subtask is finished the original computer collects the data and finishes the needed computations. A computer owner joins a Grid in order to share

its idle CPU cycles. The user also expects to use the idle CPU cycles of other users when it needs to solve a specially hard problem. Here again free riding can also be an issue, since a user may use more CPU cycles than it shares, or not share its cycles at all. Here too, the sharing of resources must be enforced so that all users may be satisfied.

Service Oriented Architectures are a new paradigm in software development. The main motivation for this approach is loose coupling of the differnt modules in the software system. These modules are defined as services and they can be accessed remotely. The access to a service is done through a series of standardized technologies, so that services can be easily used directly or grouped into more complex services in a simple way. The services can be developed by different people in different organizations. Furthermore, more than one service with the same purpose may exist and compete with one another for the attention of users. This is specially true when there exists an economic exchange for the services rendered. A user in a SOA will want to use the services with the highest quality it can get. In other words, those that will satisfy it the most. Therefore, an enforcement method would help improve the average satisfaction of users.

Finally, a Sensor Network is formed by a group of hardware sensors that have certain computing capabilities. In order to aggregate the data from these sensors they are equiped with a wireless communication device. Sensor networks may be spread through large areas and if their signal range is not long enough to reach the data gathering facility a sort of MANET is used to get the farthest nodes to send their gathered data to the central repository. The problem here is actually the opposite than in a MANET. Since all hardware is owned by the same user, it would be counterproductive to free ride. Nonetheless, the limited power source is an issue and those nodes with lower battery life should not be overused. Furthermore, those nodes that are acting defectively should be spotted so that they can be replaced. In this case enforcement is needed to make sure all nodes satisfy the needs of the gathering device.

The techniques described in this thesis are meant to be applied to distributed systems. Although the application to MAS and P2P is straightforward, these techniques could also be applied to Grid computing systems, and SOA with some modifications. On the other hand, its applicability to sensor networks and MANET is more complex, since the type of network links needed for the techniques to work ought to be long lasting, whereas MANETs and sensor networks with moving links are designed to have short lived network links.

2.2 Social Networks

Social networking applications have also boomed recently. These are part of the so called Web 2.0 wave of online sites. They are characterized by allowing users to share information with those they want to. Into this broad category one can include blogs, recommender systems, fora, and the like. These applications help in the community building process. Online communities are formed in order to

share specific types of content, such as videos, audio files, news, or even personal information. Some of these networks give users the ability to form relationships with other users to which they share specific information. This is an effective way of establishing community links.

In recommender systems, the users are allowed to give feedback on the different items available. The application uses this feedback in order to recommend content to the users. Some applications make recommendations for content that the user has not yet given feedback on. This is the common approach for movies and books. Other applications do not make this restriction, and content is recommended independently of whether the user has rated it in the past or not. This is more common in music sites or personalized radio stations.

Contact networks are a specific type of social networks in which the main purpose is to share personal information. The personal information given depends on the purpose of the site: making friends and keeping in touch with them, finding a partner, or business networking are some examples. Through contact networks you can find old friends, or business partners with specific expertise. Normally, these types of networks do not have a specific functionality that helps in assessing whether the new partner or friend can live up to the expectations. If they do, it is in the form of statistical averages over previous interactions.

A common problem with all of the social networking applications is spam. These applications have been often used by advertisers to send out information that the users do not desire be sent to them. In other cases these contact networks are used to gather information about users so that it can be used by advertisers. In the case of recommender systems, some users may try to cheat the system in order to get specific products recommended more often than they should given their real merits.

Some of these applications sought ways to lower the amount of spam, and to filter out invalid feedback when recommending items. These can be seen as enforcement techniques, since they are used to reduce the amount of undesired behaviour. Among these techniques the following ones are the most common: to allow users to block access to their content to other users; a process through which contacts are managed and classified; the ability to give feedback about the recommended items or the users with which they have interacted; and mechanisms that filter out feedback from users which try to cheat the system.

Most applications for social networking are web applications, and as such they are centralized. Nonetheless, there are examples of distributed applications that either use social networks or provide social networking capabilities. Especifically, it is P2P systems that have started to embrace social networking. There is research on using social links among peers in a distributed hash table (DHT) in order to improve the routing efficiency [Marti et al., 2004, Upadrashta, 2005].

Some file sharing applications have gone further into using social networks in their underlying structure. These P2P networks are also called Friend-to-Friend (F2F) networks, not to be confused in this context with Face-to-Face applications. Some examples are: WASTE, TurtleF2F, GNUnet, Freenet, and Darknets. Peers in a F2F network will only interact with those they have

selected as friends, thus, the probability of being cheated is lower but the scope for interaction is also limited. In [Loukos and Karatza, 2009], the scope for interaction is expanded by using an ad-hoc reputation technique through which new friends can be acquired. The technique assesses the reputation of the friends of friends and compares it to a local threshold. If the reputation is greater, the peer is added to the friends list (see Section 2.7 for more on reputation techniques).

The type of system for which the enforcement techniques have been designed in this thesis is an expanded F2F. The difference with an F2F being that a user can interact with any other user even if it is not its direct contact. The only restriction being that there must be a path in the contact network uniting the two in which all the intermediate contacts in the path allow the interaction to happen (*i.e.,* friend of a friend of a friend of a friend ...).

2.3 Norms

In systems with many users that interact with one another, the cost of establishing what makes an interaction satisfactory to the participating agents can be very high. Specially if this is done prior to each interaction. This process is as complex as the interaction and it needs planning of its own, which can be even more complex. Norms can be established that regulate what actions are to be executed so that the expectations of the agents may be satisfied while at the same time reducing the planning process.

Norms have been studied as a means of co-ordinating actions and granting access to limited resources for a long time in human societies through the study of law, philosophy and the social sciences [Fon and Parisi, 2005, Taylor, 1982]. Research in norms to co-ordinate artificial systems has been greatly influenced from those areas of knowledge [Castelfranchi, 2000]. There is a lot of research about norms, especially in the Multiagent Systems field. This research is divided into different areas: Norm representation, normative agent architectures, normative organisation architecture, properties of norms, norm spread and adoption, and norm enforcement.

There are many approaches to norm representation. Nonetheless they can be classified into the following four categories: logics, state machines, and rule-based languages. Logic-based languages where the first to be researched. These used modal logics to represent the notions of obligation and permission. The first attempt to formalise these notions was made by the Austrian philosopher Ernst Mally, but the axiomatization was not too satisfactory. It was Georg Henrik von Wright [von Wright, 1951] that satisfactorily defined the deontic logic. Deontic logic has been modified and extended to solve some of its paradoxes [Chisholm, 1963, Ryu and Lee, 1995, Tan and van der Torre, 1996, Meyer, 1988], and it has been used as the basis for some norm representation languages [Broersen et al., 2004, Sergot, 2001].

State machines define the interaction states that are normative and the actions through which the interaction state can change. An action is a norm

violation when there is no matching edge leaving the current normative state or when the state to which the action takes is not a normative state. State machine representation is preferred for simple normative systems because the graphic visualization is easy to understand. Nonetheless, if the norms are many or very complex, the visual representation becomes too crowded to be visually intuitive. Networking protocols usually use state machine representations, as do specific types of normative multiagent systems such as Electronic Institutions [Esteva et al., 2002]. In the latter case, the visual representation has a formalization in Ambient Calculus to allow for a hierarchical structuring of normative spaces. Other representations of finite state machines are possible in π-calculus [Padget and Bradford, 1999], event calculus [Marín and Sartor, 1999, Yolum and Singh, 2002], and the lightweight coordination calculus [Robertson, 2005].

Rule-based languages for representing norms have the advantage of being simple to learn to use, and they are easily broken up into modules. A normative rule defines the context and conditions under which it applies, the action which fires the rule, and the obligations, permissions, and prohibitions which are created for the rule. Many different rule-based languages have been defined which have different syntax and properties [Garcia-Camino et al., 2005, García-Camino et al., 2006, García-Camino et al., 2007, Cranefield, 2005, Craven and Sergot, 2008, Governatori, 2005]

Another aspect of normative research deals with the architecture of norm-aware agents, and with the architecture of normative organisations as a whole. The norm-aware agent architectures must have modules that allow agents to interpret norms, to reason about them, and to take norms into account in their planning process. Most approaches to normative architectures take a BDI architecture as the starting point [Rao and Georgeff, 1995]. [Castelfranchi et al., 1999] includes norms as meta goals (equivalent to desires). In [Boella and Lesmo, 2001] the agents are given the ability to choose whether the norm or obligation should be honoured. Some approaches have gone a bit further by adding new modal contexts to the original architecture. In B-DOING [Dignum et al., 2002] contexts that manage obligations, norms, and goals are added. In BOID [Broersen et al., 2001] they manage to do with just one more context that handles obligations. Finally, BDI+C [Gaertner et al., 2006] uses bridge rules to link modal contexts in a multi-context system and adds a context for commitments. Other architectures have dealt with other issues related with norms, such as conflicts that arise between them and with the agent's desires [Kollingbaum and Norman, 2003a, Gaertner et al., 2009].

Norms can be studied as an object in itself. As an object it has properties that define how it influences the MAS. There is a lot of research for classifying norms according to their properties. Early research described properties such as *usefulness* [Shoham and Tennenholtz, 1995] (*i.e.,* the norm allows all agents to achieve their goals), *minimality* [Fitoussi and Tennenholtz, 2000] (*i.e.,* the constraints the norm imposes on the allowed actions are the least for any useful norm), *simplicity* [Fitoussi and Tennenholtz, 2000] (*i.e.,* a norm which can be easily followed by agents with low computational and

sensorial capabilities). Other properties of norms that have been studied are *flexibility* [Briggs and Cook, 1995], *non-determinism* [Coen, 2000], *stability* [Tennenholtz, 1998], and *enforceability* [Boella and van der Torre, 2005]

When designing a MAS with adaptability in mind, one must design mechanisms through which new norms can emerge and through which norms can cease to exist. Emergence is a complex process [Castelfranchi et al., 2003] that consists of many parts. The ones that have been most researched are norm adoption by a single agent, and norm spread through an agent society [Walker and Wooldridge, 1995].

There are three parts to the adoption of a norm: firstly, the existence of a norm needs to be recognised; secondly, its applicability needs to be realised; and finally, the norm needs to be accepted. This norm adoption process equally applies to sets of norms. Agents need to be able to reason about joining, staying in and leaving a social group, taking into account the norms that regulate the group and the agents' goals. One such agent model is described in [López y López and Luck, 2004]. When considering norm adoption, the agent must be able to cope with inconsistencies. The new norm may oblige for actions that are contrary to the agents goals and beliefs. These conflicts have been addressed through modules in the agent's architecture [Kollingbaum and Norman, 2003b], either through meta-ordering of norms [García-Camino et al., 2006, Sartor, 1991, Sartor, 1992], through relaxation of the axioms of standard deontic logic (SDL) [Cholvy and Cuppens, 1995], through variable instantiation graphs [Kollingbaum and Norman, 2004, Vasconcelos et al., 2007, Gaertner et al., 2007], or through the maximisation of the coherence of norm subsets [Joseph et al., cess].

Adoption is to a single agent, what spread is to the multiagent system. The two main approaches to researching the spreading of norms in AI are evolutive or learning. In the evolutive approach agents with low success leave the system (*i.e.,* die) and agents with high success are copied (*i.e.,* reproduce) and some mutation is possible [Axelrod, 1986]. In the learning approach agents that are less successful copy the strategy of the agents that are more successful [Savarimuthu et al., 2007]. Both approaches look for the conditions which guarantee that the strategies that survive are those that follow the norms. Some of this research deals with structured MAS. In [Kittock, 1994] the influence of simple structures such as regular graphs, trees, and hierarchies on the spread of norms was studied. Later work studied the influence of complex structures possessing free-scale or small-world properties that can be found in many natural systems [Delgado, 2002]. Finally, in [Pujol et al., 2005] the relationship between norm emergence and other graph parameters such as its clustering factor and diameter is studied.

The norms that regulate the interactions in such systems may be defined by the system designer, or they can emerge through interaction between the users. In the first case, it is the system designer that is interested in having these norms abided with, and she will provide the mechanisms through which the norms will be enforced. On the other hand, when the norms emerge because they are useful to the users, the system ought to have mechanisms through which these norms

may be enforced by the users themselves.

This thesis deals with enforcement techniques. One of the scenarios in which these techniques are applied is a normative system were the expected behaviour is predefined and shared by all the participants in the system. The effect of the enforcement techniques to the mentioned scenario is studied in Chapter 4.

2.4 Enforcement

Norm enforcement in multiagent systems is done under one of two premises: a) the designer can control the actions agents realise in the system and can stop non-normative actions before they take place, or b) no one except the agent can control its actions, and enforcement must be executed after actions have taken place through the use of sanctions or rewards. Systems such as S-$MOISE^+$ [Hübner et al., 2006] or Ameli [Esteva et al., 2004] are designed with the first premise in mind. All interactions between agents are mediated by some trusted component implemented by the system designer, which verifies that actions are normative. Other systems use model checking techniques to verify that the agent code will fulfil all of the system's norms [Ågotnes et al., 2007, Minsky, 1991b]. For those systems where non-normative actions cannot be prevented, actions from one agent will be responded to with actions from other agents so as to make normative behaviour better. These actions are called incentives.

According to the previous two premises, enforcement techniques can be classified into two main categories: control-based and incentive-based enforcement techniques. The main goal of control-based techniques is to bring about the conditions in which deviant behaviour has no effect on the rest of the society. When using incentives you allow agents to break the norms, but sanctions or rewards are given to make deviant behaviour less attractive than conforming behaviour.

2.4.1 Control-based Enforcement (CBE)

As said earlier, CBE, tries to bring about the conditions so that deviation from the norm has no effect on the rest of the society. Early work on Law Governed Systems (LGS) by Minsky in [Minsky, 1991c] proposed a system with two modes of enforcement; *by interception* and *by compilation*. In enforcement by compilation the enforcer analyzes the source code of all the new components trying to join the system. The enforcer studies the component's code to evaluate whether the messages that can be sent by it conform to the system's laws. If they do not conform, they are not allowed to join. Enforcement by interception requires having a component at run-time intercepting all the messages sent by the rest of the components, and dismisses those that do not conform to the laws. All software components are linked to this centralized *law enforcer*. In order for two regular components to interact with each other, their messages have to be sent to the law enforcing component, which parses them and verifies their conformance to the law. The law enforcer then routes the conforming messages

to the appropriate component, and dismisses the deviant messages. This way, all the interactions taking place in the system forcefully abide by its norms.

LGS are easy to implement and deploy, but have some drawbacks. Firstly, enforcement by compilation can only detect violations that can be caused by messages sent by agents that do not conform to the specified laws. It cannot detect if agents violate their obligations by not sending messages when these are required. This problem is solved in the interception enforcement mode, which deals with obligations by having the law enforcer remember each agent's obligations and what is termed the *control state*. In case an agent does not send a message it is obliged to send, the law enforcer will send it for him (this implies that the protocols that can be specified are rather limited). However, the interception mode also has its drawbacks, mainly the fact that law enforcement is centralized. This means that it is not scalable, there is a limit to the amount of interactions a law enforcer component can verify. Furthermore, if the overhead of computing the conformance to the laws is large enough, the monitoring capacity will be small and the law enforcement may become the bottleneck.

In [Minsky, 1991a] Minsky proposes a distributed enforcement technique for an open system. The new approach consists of decentralizing the enforcement task to many enforcer components. Each component joining the system is provided with a law enforcer that is placed between the component and the network, routing all its messages. Minsky calls these components *controllers*. Whenever an agent wants to send a message to another agent, its controller will intercept the message and verify if it complies with the norms. If the message is norm-compliant, the controller sends the message across the network to the receiving agent's controller, which will also verify that the message is norm-compliant before delivering it to the receiving agent. Under this framework, each controller must keep a copy of its controlled agent's state, in order to verify that the norms are being followed. Furthermore, the controller agents know what obligations must be fulfilled by the agents they control. If the controller realizes that an obligation is not being fulfilled by the controlled agent, it will send the appropriate message itself. Certainly, this can only be done when the controller knows all the parameters needed to create the message. This fact, reduces the amount of unfulfilled obligations that the controller can handle.

In [Minsky and Rozenshtein, 1988] Minsky and Rozenshtein implemented Darwin, a software development environment for law-governed systems based on the previous enforcement modes. Later, in [Esteva et al., 2001] a more powerful formalization allowed a multi-agent system to enforce more complex protocols; they were termed *Electronic Institutions*. In a way Electronic Institutions are an extension to Minsky's enforcement by interception model. Each agent joining the institution will have what is called a *governor* as a proxy to all its communications, it is the conceptual equivalent to a controller. Furthermore, in Electronic Institutions, protocols can be specified by linked sub-protocols. Each agent can move from one subprotocol to the next by joining and leaving *scenes*, hence the need for another type of agent that manages the scene's state, the *scene manager*. In Electronic Institutions, though, governors cannot deal with unfulfilled obligations, therefore norms must specify what must be done

in such cases. Grizard in [Grizard et al., 2007] adds reputation to the control process by having controllers calculate the reputation of the agents they control, and sending this information to other controllers and agents as part of the interaction process.

2.4.2 Incentive-based Enforcement (IBE)

In some cases intercepting all the communications trying to spot norm violations may imply too much overhead (as in the case of real-time scenarios), or it may even be impossible (as in peer-to-peer systems). Furthermore, it is complex to enforce obligations using CBE. For these situations, incentive-based enforcement can be used. The main problem with IBE is that it cannot ensure that all actions will be lawful. Its aim is to minimize deviant behaviour. Incentives can be divided into two types; positive (when the norm is followed), or negative (when it is violated). Negative incentives are also known as sanctions, and positive incentives as rewards [Oliver, 1980].

Incentives can be established using different types of goods. In the case of electronic agents, three types of goods have been used as incentives: utility, reputation, and access to resources (e.g., CPU, network...). IBE can be self-enforced or enforced by a third party. In the case of self-enforcement, an agent A interacting with another agent B will use incentives so that B abides by the norms. In the case of third party enforcement, a third-party agent X is added to the interaction, and depending on A and B's actions, X will apply the appropriate incentives. Furthermore, [Yarbrough and Yarbrough, 1999] argues that self-enforcement can only be brought about if there exist some kind of linkages between agents. Linkages are of three types: Intertemporal, inter-issue, and inter-actor. In which case lowering deviant behaviour equates to maintaining the boundaries between those agents that are in the respectable group (insiders) and the rest (outsiders).

In most literature the enforcement problem is dealt with from a game theoretic perspective (see section 2.6) in which the agents interacting play a game that has different utility outcomes depending on their choice of action. A norm establishes which outcomes are valid, although an agent can choose whatever action it likes. Following the game-theoretic approach and using utility as the only possible incentive, third-party enforcement can be modeled as a three dimensional matrix, and self-enforcement will necessarily take place in an iterated game, since an agent can only wait for the next interaction to apply its incentive. For example, if two agents interact to exchange some goods, and one of them does not fulfill its part of the deal, the aggravated agent can only sanction the violating agent in a subsequent interaction, which is called *reciprocity* [Axelrod, 1985]. In the case of third-party enforcement (the enforcer being the bank), the norm-violating agent could have some utility removed by the third-party enforcer. Of course this is only possible if the third-party enforcer agent is given access to the utility of the interacting agent, which could be accomplished as a prerequisite to joining the system.

When using reputation or access to resources as the incentive, there are ac-

tually two games being played in parallel. First of all there is a utility game in the game-theoretic sense, as discussed in the previous paragraph. But on the other hand, there is an incentive game where the reputation and resource incentives are applied. The utility game influences how the incentive game is played, but also the incentive game has an effect on whether the agent will be able to play the utility game. For example, in the case of using reputation as an incentive, an agent A interacting with a norm violator B may gossip about the interaction's outcome to other agents, which in turn will lower B's reputation. This affects B's capacity of interacting in the utility game, because agents try to avoid interactions with agents who have a low reputation [Castelfranchi et al., 1998, Hales, 2002, Younger, 2005, Younger, 2004]. The reputation could be calculated through a distributed mechanism, such as in [Sabater and Sierra, 2001, Ramchurn et al., 2004], in which case it would be self-enfocement. A third party enforcement setting, implies a centralized service run by an agent that gathers all the gossip, calculates each agent's reputation, and replies to queries by agents about other agent's reputation. When using access to resources as an incentive, an agent may lower the amount of network access to an agent deviating from the norms, or even forbid it. In which case, the violating agent may not be able to achieve its utility goals [Perreau de Pinninck et al., 2008b].

Another technique is sometimes used before the other two are applied: persuasion. In this stage the agents communicate with each other in order to deter deviations from the norm. This can be achieved through one of the following two methods:

- Argumentation — by which an agent argues with the other agent in order to convince it not to deviate from the norms.
- Threats or Offers — in which the agent lets the other agent know what actions will be brought about in the incentive section. In the case of a threat, one agent lets the other one know what sanction will be applied in the case that the other agent deviates from the norm. On the other hand, an offer is when one agent tells the other what reward it will give if the agent abides by the norm. These two may not be mutually exclusive [Taylor, 1982].

Most articles dealing with IBE are based on simulations. The scenarios used to simulate normative societies with enforcement capabilities are the same as those used for simulating the emergence of norms. Stochastic games such as Axelrod's Iterated Prisoner's Dilemma (IPD) and resource gathering scenarios such as Conte and Castelfranchi's food gathering game [Conte and Castelfranchi, 1995].

The enforcement techniques described in this thesis fall under the IBE category. Interactions are prevented from happening not because it is known that they will not be satisfactory or they will incur norm violations, but because the prevention is a sanction to one of the interested agents for a previous interaction that was not satisfactory or was a norm violation. The mechanism through

which these sanctions are applied is based on getting agents to need others in order to interact. Therefore, sanctions consist on not co-operating with an agent that needs the co-operation in order to interact.

2.5 Violation Detection

In order to enforce norms, the system or the agents doing this enforcement must have the ability to detect norm violations. In order for norm violations to be detectable, the actions of agents must be observable either by other agents or by the system. In normative multiagent execution frameworks, such as S-MOISE or Ameli, actions by agents are observed by the system agents because external agents are assigned a system agent that will route and verify all their messages. This is the mechanism used for CBE techniques in which a mechanism is provided so that the content of interaction among agents can be scrutinised by organisational agents in order to detect when the interaction reaches illegal states [Aldewereld et al., 2006, Vazquez-Salceda et al., 2004]. When not all actions can be observed, the system or other agents have to detect violations through the subset of actions they have observed.

Actions can be observed by one of the following methods. The first method is by observing the changes in the environment. This mechanism assumes that the MAS platform provides an environment that can be changed by the agents, and that agents can know which changes have been done by whom. One way to achieve this is by overhearing the messages sent by agents, even though the messages are not sent to the overhearing agent. The second method is by observing the actions affecting the detecting agent as part of an interaction. The last method is through gossip, where an agent informs other agents that a norm has been violated. The problem with gossip is that the source must be trusted to be saying the truth.

Overhearing is quickly gaining attention as a general method for monitoring open distributed MAS. Overhearing is a technique through which agents listen on conversations on which they are not taking an active part in order to provide different kind of services. Among these services, we find enhanced situational awareness for pilots [Novick and Ward, 1993], organizational knowledge of roles performed by agents [Legras, 2002, Rossi and Busetta, 2004, Rossi and Busetta, 2005], recommendation for interesting roles to be played [Cabri et al., 2006], monitoring progress in task execution, plan recognition [Kaminka et al., 2002], inconsistency detection and performance analysis for task execution [Rossi and Busetta, 2004, Rossi and Busetta, 2005], issuing alerts and notifications [Cranefield, 2007, Rossi and Busetta, 2004, Rossi and Busetta, 2005], and reducing the communication costs by filtering messages that do not conform to the protocol norms [Perreau de Pinninck et al., 2008a]. Furthermore, MANETs that use trust-based routing schemes to avoid free-riders have used overhearing of communications to insure that messages are re-routed. Nonetheless, they do not use the overhearing techniques mentioned above.

This thesis does not cover the mechanism through which an agent detects norm violations or how it decides if an interaction is satisfactory. It is assumed that agents have this capability. The core of the work in the thesis is based on a system in which it is only the interaction participants that receive the contents of an interaction. Therefore, overhearing cannot be used as a violation detection tool. Nonetheless, the data from interaction bootstrap is used in Chapter 5 to certify that interactions have indeed taken place, thus validating the legitimacy of complaints.

2.6 Game Theoretic Research

The first research on enforcement via agent simulations [Axelrod, 1985] sought ways to ensure co-operation in situations where agents had high incentives to avoid co-operation. A utilitarian approach was taken in that research by modelling interactions amongst agents through an iterated prisoner's dilemma (IPD). In that approach interactions could not be avoided and self-enforcement was achieved through reciprocation. Therefore, an agent that did not co-operate did not receive reciprocated co-operation in the future. This was termed *the shadow of the future*. The problem with limiting the interaction in an IPD, where interactions cannot be avoided, is that an agent is forced to stop co-operating in order to sanction a noncooperator. This can end up in a spiral of noncooperation. If the norm is to co-operate, the norm must be broken in order to sanction. Later research solved this problem by modifying the original game of prisoner's dilemma and adding an enforcement stage where agents could decide to sanction those that did not co-operate [Axelrod, 1986]. The utilitarian point of view to enforcement showed that when norm-violators are punished violating the norm was not the utility maximising strategy [Castelfranchi et al., 1998]. Nonetheless, if applying sanctions has a cost of its own, then agents may not want to sanction others, which brings about a free-riding problem [Carpenter et al., 2004]. In such cases, adding a norm saying that agents ought to enforce norms does not solve the problem. It is just shifted up a level since agents may not enforce the enforcement norm, bringing about an infinite regression.

Mechanism design also studies how to get agents to act in the way that the designer of the system wants them to. Through mechanism design the rules of the games are designed so that a specific outcome can be achieved. Incentives are provided, from a game-theoretic point of view, for utility-maximising agents to present specific properties when interacting with others. Mechanism design could be categorized loosely under the CBE class of enforcement techniques. Some of the properties sought through mechanism design are very interesting from an enforcement perspective, such as the truthfulness property through which it is ensured that it is in the best interest of the participating agents in the protocol to be truthful in their evaluations. Another interesting property that can be achieved through mechanism design is the maximization of certain values, such as social welfare, or bud-

get balance. In order to really understand mechanism design one has to get technical. The interested reader is referred to the excellent introductions in [Jackson, 2003, Parkes, 2001, Maskin and Sjöström, 2002], and the most recent survey on mechanism design for computer scientists in [Nisan, 2007].

The issue with mechanism design, being a game-theoretic approach at heart, is that agents are assumed to be able to give a value to each of the outcomes of the interaction with other agents through the protocol. The main difference with the approach presented in this thesis is that through mechanism design an interaction protocol is engineered in a way that it enforces that agents will act in the way that the engineer desires because of the properties of the protocol. Whereas in our approach the interacting agents are given enforcement capabilities so that they can decide for themselves which are the proper outcomes of an interaction and enforce them.

2.7 Reputation Systems

When CBE techniques are not possible, an agent has to be careful when choosing its interaction partners. In order to choose, agents must model the behaviour of others so as to know if they would be satisfactory interaction partners. Modelling of this sort has been studied in what is termed trust management systems (TMS). There is a lot of research into designing TMS that can predict the satisfaction in interactions with other agents. These systems have two types of inputs: the history of past interactions of the agent making the assessment, and feedback or gossip about interactions of other agents. Gossiping is a type of third-party enforcement technique. By spreading knowledge about unsatisfactory interactions, the unsatisfied agent can convince other agents not to interact with a specific agent.

TMS can be implemented by centralising the algorithms that calculate reputations (*e.g.,* eBay). Centralisation poses two problems: firstly, the system where the calculations are centralised becomes the bottleneck in large environments; and secondly, the centralized system must be trustworthy otherwise it could take advantage of everyone else. These problems are tackled by the implementation of a decentralized approach, which has problems of its own. Decentralized approaches [Aberer and Despotovic, 2001, Damiani et al., 2002, Kamvar et al., 2003, Sabater and Sierra, 2001, Xiong et al., 2004] must implement distributed information gathering systems that are efficient and robust, neither of which is straightforward. The more robust to information loss a distributed system is made, the less likely it is that it will remain efficient, and vice-versa.

Where feedback or gossip is involved, the TMS must be able to filter malicious feedback, otherwise agents can use the feedback mechanism to artificially change their own or other agents' reputation. Known malicious uses of feedback are: badmouthing, ballot stuffing, and colluding. Badmouthing consists of an agent giving negative feedback about another agent even though it was satisfied with the interaction. An agent may be moved to badmouth another

if they are competitors for a specific service. Through badmouthing the agent may indirectly get more clients, since some client agents will be deterred from interacting with the badmouthed agent and may end up interacting with the badmouther. Ballot stuffing consists of giving more feedback than actually exists, this can be done by making up fake interactions and reporting feedback on them. Ballot stuffing of this sort can be easily avoided via cryptograpy. Finally, agents can collude by faking interactions and giving good feedback on those fake interactions so as to increase their overall reputation.

There are different approaches in the literature that tackle malicious feedback in different ways. Eigenvectors are used in [Kamvar et al., 2003] to filter out badmouthing and ballot stuffing. Nonetheless, the proposed solution relies on a set of pre-trusted peers in order to avoid collusion attacks. Another approach is to take into account the credibility of the feedback origin by calculating the difference in average ratings between the feedback originator and the agent assessing the trust [Aberer and Despotovic, 2001, Xiong et al., 2004]. This approach does not need to rely on pre-trusted peers to prevent collusion attacks.

Some of the trust and reputation management systems in the literature use a distributed hashtable (DHT) to store the feedback given by other agents [Kamvar et al., 2003, Aberer and Despotovic, 2001, Xiong et al., 2004]. The DHT in such cases is formed by the whole set of interacting agents. Each agent takes care of a subpart of the DHT and manages a subset of the feedback stored on it. Therefore, malicious agents may try to subvert such a system by deleting the feedbacks it manages in order to suit its needs. Each peer in a DHT should handle a subset of the key-value pairs. The key-value pairs assigned to a peer depend on the hash value associated to the peer's identifier, thus distributing the keys through the different participants in the DHT. If a user were to create enough fake peer identities, it could easily control those key-value pairs that interest it. One way in which this variant of the Sybil attack [Douceur, 2002] is tackled is by replicating the key-value pairs in different agent-nodes [Xiong et al., 2004]. When the value for a key is retrieved from the DHT, it has to be returned by all the agent-nodes that manage the given key. If one of the agents repeatedly fails to return the correct feedback, it is known to be trying to subvert the system.

Some applications may require agents to have many different expertise, since there will be many different types of interaction. Therefore, the TMS must also take into account the context of the interactions when assessing the trust or reputation on another agent. In [Sabater and Sierra, 2001, Sierra and Debenham, 2005] special care has been taken in developing a model through which reputation in different contexts can be managed. The problem of multiple contexts is that the higher the number of contexts, the smaller the quantity of feedback for each one. Therefore, multi-context TMS must establish similarity between contexts, which is a complex topic in itself.

Agents can also try to subvert the TMS by displaying a dynamic personality through which they first gain reputation by acting satisfactorily and then milking the gained reputation by acting unsatisfactorily. Some TMS handle

dynamic personality attacks by either measuring the entropy in the feedback [Sierra and Debenham, 2005] or by reducing the time window of the feedback taken into account when the agent's reputation decreases [Xiong et al., 2004].

Another way to subvert the TMS is via whitewashing. In this case an agent with low reputation will change its identity in order to shed its former reputation. This is useful when the reputation of an unknown agent is higher than the current one. Whitewashing attacks are only viable when users can change their online identities easily. Some TMS researchers avoid the issue by assuming that identities cannot be changed. When such an assumption is not reasonable, a potential solution is to add a cost to all newcomers alike. In [Feldman et al., 2004b] an analytical game theoretical study shows that if the cost associated to getting a new identity is higher than the utility gained by cheating, such mechanisms effectively tackle whitewashing attacks. Nonetheless, in that work they show that such costs degrade the systems efficiency when turnover is high, because they assume that it is impossible to differentiate a real newcomer from a whitewasher. In [Sun et al., 2005], a mechanism with similar ideas is used to tackle whitewashing attacks. This is an adaptive mechanism that assesses the trust on a newcomer based on the rate of newcomers that cheat in a time interval. In order to restrict the range of the effect of whitewashers on newcomers they add a grouping mechanism.

Reputation mechanisms are necessary companions to the enforcement mechanisms for scenarios with uncertainty. When satisfaction is not a black and white issue, the use of a reputation mechanism to model the behaviour of others helps the agent decide whether or not to sanction, and when it is subjective. Chapter 5 deals with scenarios where uncertainty is the norm, and reputation mechanisms are big players in those scenarios.

2.8 Currency Systems

Human societies have been using currency as a common way to measure utility. This currency in turn is used as a means to create incentives, either as sanctions or rewards. The currency systems used by humans are usually managed by the States, and thus are somewhat centralized. There are computer applications that are used for trading services and goods (*e.g.,* eBay, iTunes, or Amazon) and that use these currencies. Nonetheless, there have been efforts to establish online currencies for micro-payments. These online currencies are used by electronic agents to pay for the services of other agents. Agents can offer their services in exchange for the currency, which they can later on exchange for other services. Micro-payment systems have been devised mainly to stop free-riding, but they could also be used as an enforcement mechanism to achieve any desired conduct.

Micro-payment approaches share similar issues with TMS. They can also be implemented in a centralized fashion, in which it is one agent that handles all the transactions and keeps the accounting [Vishnumurthy et al., 2003]. Here again we have the bottleneck and trust issues with the accounting agent. In this case, implementing a decentralized currency management system is also tricky.

The idea is that any agent can create currency, which can then be transferred to other agents as a payment [Garcia and Hoepman, 2005]. Distributed currency management systems must also take fraud into account, such as double spending or fake coins. In order to avoid agents from creating fake coins, it must be a computationally costly task to create a valid coin, but verifying whether a coin is fake or duplicated should be inexpensive. In [Garcia and Hoepman, 2005] this is achieved by having a coin represent a hash function match.

The main problem with micro-payment systems is that the services being paid for are so inexpensive that the overhead from the micro-payment is too high. Especially when specific techniques have to be used to avoid fraud. Ripple [Fugger, 2009] is a simple currency scheme based on an underlying social network through which credit is given to trusted contacts, which could be used as inspiration for simpler currency systems so that the overhead is not too great for the price of the services being paid for.

Distributed currency systems could also be a companion to the enforcement mechanisms proposed in this thesis. Although some research has been done to check how well currency systems would work with the enforcement mechanisms by the author, it has not been included into this thesis. The use of currency systems could be used to tackle free-riding, which has been shown to be a problem in many P2P and MANET systems.

This chapter has described the state of the art in the research areas related to this thesis. The work in this thesis provides new enforcement techniques for distributed systems. These enforcement techniques are power-based and use information from an underlying social network. On one hand, the work in Chapter 4 is closely related to norm research since its scenario is a normative multiagent system. The research that ultimately led to the work in that chapter was based on game theory and the iterated prisoner's dilemma. On the other hand, the work in Chapter 5 deals with subjective satisfaction measures, thus, reputation mechanisms are used to model the probability of satisfaction. However, a currency-based mechanism could have been used for the same purpose. In both of these chapters, there is no need for complicated violation detection techniques, since there is no relation between different joint actions. Nonetheless, if such relations existed, violation detection techniques would be essential. Before going into the main contributions in Chapters 4 and 5, Chapter 3 presents the methodology that has been used for the experimental results.

Chapter 3

Experimental Methodology

In this chapter we present the methodology that has been used to test different hypotheses by means of experiments throughout the thesis. By dedicating a chapter to the methodology, the chapters where experiments are actually described are leaner, which makes them more readable. This chapter is meant as reference to the reader in case she needs to remember how the experiments in the core chapters have been executed.

The experiments consist of computer simulations of the models defined in Chapters 4 and 5. The models have many variables, with many possible values each. Nonetheless, each experiment deals with different ranges of values of the variables depending on the nature of the hypothesis to be tested (the types of variables and a list of all of them is given in Table 3.1). Simulations are executed with the purpose of extracting data, which comes in the form of measurements taken throughout the simulation or at the end of it. This data is then analysed to support or discard the hypothesis.

Since the models have many variables, the best way to test the hypotheses is through factorial experiments. These experiments are especially well suited when there are two or more factors (important variables). Each of these factors having a discrete set of input values, these values can be a selected subset of all the values available. The idea is to run the simulations with all possible combinations of these selected values accross all the factors. Factorial experiment designs can also be called fully-crossed designs, but the former name will be used throughout the thesis. Factorial experiment designs allow the experimenter to test the effects of specific factors on the output measurements, as well as the effects of interactions between factors, while making sure that the other factors did not have a role in those effects.

We have implemented a simulation environment in Java. This environment allows the researcher to parameterize the variables of the model. The simulation tool can also take specific measurements at different intervals. Also, simulation batches can be executed for factorial experiments. The simulation tool has some random effects, which are not accounted for in order to reduce the number of factors, such as the order of interaction, the randomization of some agent

strategy parameters, or the network graph. In such cases, several simulations with the same set of input variable values in a factorial design are run in order to make up for the random effect.

Once the simulations are all run and the data has been gathered, then the statistical analysis begins. We use an analysis of variance (ANOVA) to test whether there is an effect from specific input variables to specific output measurements. An ANOVA works best when the input data has the following characteristics: independence, normality, and homoscedasticity. Independence means that values of one variable make neither more nor less probable values of another variable. Normality means that the measured values for any input variable value combination have a normal distribution. Finally, a sequence or a vector of random variables is homoscedastic if all random variables in the sequence or vector have the same finite variance.

ANOVA tends to be robust with measured data that is not completely normal or doesn't have equal variances. Nonetheless, we run a quantile-quantile plot (QQ plot) through which we test the experimental data for normality. After passing the QQ plot test, we run a Levene's test, which verifies homoscedascisity of the data. Both of these tests are not completely necessary, since ANOVA is robust to non-normality and non-homoscedascisity to a certain degree. On the other hand, ANOVA is not robust to dependent variables, and we make sure there are none through a correct experiment design.

Once we have checked for normality and homoscedascisity satisfactorily, we run the ANOVA to test the effect of specific input variables or groups of input variables on the output measurements. The ANOVA analysis returns the significance of a variable or group or variables as a measurement of the probability with which they had an effect on the output measurement. The convention is that, if the significance is over 95%, then an effect exists.

For those experiments where we want to know the specific effect the input variable had on a specific measurement, we run a Tukey test for the significant factors. The Tukey test is a single-step multiple comparison procedure to find which means are significantly different from one another. By comparing the means for the different input values of the given factor one checks which input values have significantly smaller or larger output measures.

All the statistical analysis tests (QQ plot, Levene, ANOVA, and Tukey) have been run using the R statistical package [R-Project, 2009]. This is a free software replacement for the well-known S statistical package. The language used for programming is the same for both systems.

3.1 Variables

All the simulations and experiments follow the models presented in Sections 4.2 and 5.1. These models can be parameterised in many ways. All of these parameters are in fact the input variables (or factors) taken into account for the statistical analysis. In this section we describe some of the main variables set in most simulations and how they are used in the simulations.

Name	Type
Number of Agents	2, ∞
Number of Rounds	1, ∞
Network Topology	Small-world, Scale-free, Tree, Ring, Random
Interaction ordering seed	1, ∞
Random seed	1, ∞

Table 3.1: Simulation Variables

Table 3.1 describes the variables in the simulations for a specific model. In this case the number of agents is a natural number that must be at least 2. In order to test different values for this variable we get numbers from a power distribution: 2^4, 2^5, 2^6, 2^7, 2^8, and so on. The number of rounds is also a natural number, which defines the total number of time steps for which the simulation will be executed.

The network topology establishes the general makeup of the contact network. Different topologies have been used in the simulations, some of these topologies are significant because they have characteristics which are found in real world contact networks, such as scale free and small world topologies. Other topologies, such as trees and random networks have also been used in some experiments.

In each time step each agent is given the chance to start an interaction, which may end up in a joint action with another agent. The order in which the agents are given the chance depends on a random variable which is given by the interaction ordering seed. The random seed is used to generate the pseudo random values that are used to generate the network graph of the specified topology, the cheating probability values, and other inputs that require a random value.

3.2 Measurements

The simulation tool takes measurements at specific time intervals, by default this interval is set to 10 rounds. Depending on the experiment at hand, different measurements will be taken. All measurements are taken together at each interval. These measurements can be taken for all agents as a whole or just for those agents of a specific type. In this section we present the different measures that have been taken.

Table 3.2 describes the different output measurements of the simulations for an experiment. The number of messages is the measure of how many low level messages have been sent by all agents. These messages include request messages, acknowledgement messages, complaint messages, and feedback query and answering messages. Messages from low level protocols and routing table maintenance messages are not taken into account.

The number of joint actions is the total number of joint actions that have

Name	Type
Round	$\mathbb{N}$
Number of Messages	$\mathbb{N}$
Number of Joint Actions	$\mathbb{N}$
Number of Satisfactory Joint Actions	$\mathbb{N}$
Number of Complaints	$\mathbb{N}$
Number of Attempted Joint Actions	$\mathbb{N}$

Table 3.2: Simulation Measurements

taken place. The number of complaints is the total number of complaints filed by agents. A satisfactory joint action is a joint action for which none of the participants have complained. None of the previous three values can be calculated from the other two because a non satisfactory joint action could involve one or two complaints. Therefore, the three have to be measured at runtime if they are needed for the experiment outcome. The number of attempted joint actions is the total number of interaction protocols started, independently of whether they ended up in a joint action.

Other derived metrics may be useful for the experimental result analysis, such as the percentage of successful joint action attempts, or the percentage of satisfactory joint actions. This can be easily calculated given the previously mentioned measurements. For example, the percentage of satisfactory joint actions is calculated through the following formula,

$$\frac{\#SatisfactoryJointActions}{\#JointActions}$$

The derived metrics used for the specific experiments will be explained at the experiment sections of the main chapters.

3.3 Experiment design and results

In the chapters where experiments are presented, a table is given to represent the experiment design. This table consists of three parts: the first one contains the input variables of the simulations, the second one lists the measurements taken throughout the experiments, and the third part describes the integrity constraints for the input variable values. The first part of the table contains one row for each tested variable, or factor. When listing the variables that depend the agent strategies they will be grouped by agent type. The table has two columns, the first column contains variable names, the second contains the actual values used in the simulations. As for the second part, there is one row for each measured value. The first column names the measured variable, and the second gives the measurement type. Finally the third part of the table contains one integrity constraint per row. Each row has two columns, one for the constraint name and one for the constraint definition.

Input Variables	**Values**
Agents	32, 64, 128, 256
Rounds	10, 20, ..., 200
Topology	Small world, Scale Free
Cheater	0.1,0.2,0.3,0.4,0.5
- Cheating probability	0.8, 1.0
- Colluding	True, False
Collaborator	0.5,0.6,0.7,0.8,0.9
- Blocking	True, False
- Informing	True, False
Measurements	**Type**
Messages	$\mathbb{N}$
Satisfaction	$\mathbb{R}$
Integrity constraints	**Formula**
Full Partitioning	Cheater + Collaborator = 1.0

Table 3.3: Example of an experiment description table.

Inputs	**Outputs**	α	**Relationship**
Agents	Messages	0.98	Higher number of agents have higher number of messages
% Cheaters	Satisfaction	0.95	Higher percentage of cheater agents make for lower satisfaction.
Blocking	Satisfaction (Collaborators)	0.99	When blocking is used by collaborators, the satisfaction of collaborators is increased.

Table 3.4: Example of an experiment description table.

Table 3.3 gives an example of an experiment design table. The first part shows that there will be two types of agents: cheaters and collaborators. The percentage of the population that is a cheater goes from 10 to 50, and the percentage of collaborators is dependent on the number of cheaters. Therefore, it cannot be used as an input variable for the ANOVA. Each agent type has different strategy settings, which are shown below the agent type name.

The experiment results are also given in a table. Each row in the table represents a potential cause-effect relationship. The first column in the table gives the variable, or group of variables, that can have a potential effect. The second column shows the measured variable over which there might be an effect (the contents in parenthesis define over which agent types the measurements hold). The third column gives the significance value. If the value is over 0.95, then a correlation exists. The last column explains the relationship between the input and output, if any.

Table 3.4 gives an example of what a results table would look like. There are three results showing three relations between groups of input variables and

groups of output measures. All the variables and measures are those present at Table 3.3. The first row shows that the relation between the number of agents and the number of messages is highly significant, and that the form of the relationship is given in the last column.

Chapter 4

Ostracism

In a multiagent system where norms are used to regulate the actions agents ought to execute, some agents may decide not to abide by the norms if this can benefit them. Norm enforcement mechanisms are designed to counteract these benefits and thus the motives for not abiding by the norms. In this chapter we propose distributed mechanisms through which agents that do not abide by the norms can be ostracised by their peers. An ostracised agent cannot interact anymore and looses all benefits from future interactions. We describe a model for multiagent systems structured as networks of agents, and a behavioural model for the agents in such systems. Furthermore, we provide analytical results which show that there exists an upper bound to the number of potential norm violations when all the agents exhibit certain behaviours. We also provide experimental results showing that both stricter enforcement behaviours and larger percentage of agents exhibiting these behaviours reduce the number of norm violations, and that the network topology influences the number of norm violations. These experiments have been executed under varying scenarios with different values for the number of agents, percentage of enforcers, percentage of violators, network topology, and agent behaviours. Finally, we give examples of applications where the enforcement techniques we provide could be used.

4.1 Introduction

Multiagent systems (MAS) consist of groups of agents that interact with one another with the purpose of achieving their individual goals. In order for multiagent systems to be viable, the agents involved should have a better chance of achieving their goals by interacting with others in the MAS than by trying to achieve them on their own. When interacting with other autonomous entities, which are potentially selfish, planning can be a complex task. For situations where planning is cumbersome, norms may be established that restrict the set of valid actions, thus simplifying the planning process. However, an autonomous agent can choose whether to follow the norms or not.

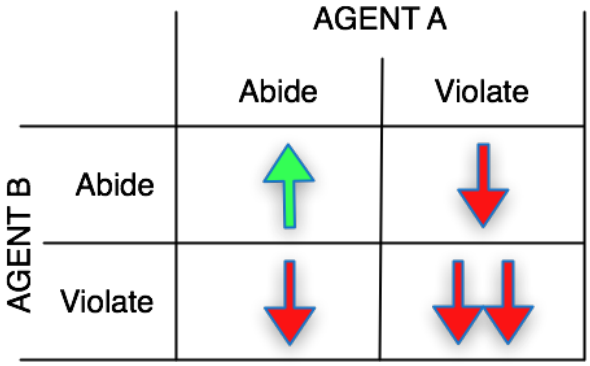

Figure 4.1: Global outcomes of interactions

It is mainly in the interest of the designer of norms in multiagent systems that agents abide by them. In Figure 4.1 we can see a table showing the qualitative gain achieved by the designer (or by the community as a whole) by interactions among pairs of agents of the system, which are defined as joint actions. Agents have two main choices: to abide by the norm or to violate it. The most desired interactions are those in which both agents abide, this brings about a positive outcome. On the other hand, interactions where any agent violates the norm are not desirable, being those interactions where both agents violate the norm the most undesirable of all (see Figure 4.1). Nonetheless, some individual agents may get more satisfaction out of executing illegal actions themselves, otherwise there would be no need for enforcement. The situation where norms are beneficial when all agents abide, but where agents also have an incentive to break the norm is the one where the normative behaviour is hardest to achieve.

The purpose of this chapter is to find behavioural properties of the agents that are better suited to achieve norm compliance at the global level. We examine those behavioural properties that can serve as distributed norm enforcement techniques in the MAS we study. We do not aim to study the internals of agents in order to determine what might motivate them. We treat agents as black boxes whose internals we do not have access to and we study the outcome in norm abidance when different agents with different behaviours interact.

Ostracism is the exclusion by general consent from common privilege or social acceptance.[1] Ostracism is a peer norm enforcement technique, *i.e.,* a technique applied among equals to enforce norms. In the approach described in this chapter there is no co-ordinated action by the community to ostracise, it is a gradual process by which a violator agent is removed through the actions

[1] Ostracism was first practised by the ancient Greeks as a method of temporary banishment by popular vote without trial or special accusation. The way ostracism was decided in Athens was by casting a vote in pieces of broken pottery called ostraka. If enough votes were cast, the person with the highest number of votes was forced into exile for ten years, after which he was allowed to return without loss of status. If he tried to return before that he would face a death sentence.

of its peers. Furthermore, there is no explicit expiration time for ostracism, it is individual agents who choose whether to readmit the violator or not. The inspiration for our approach comes from the network security area, where firewalls are the most commonly used tool to avoid undesirable interactions. Such firewalls are managed by technicians which set up the rules under which they operate. These rules define which communications they allow and which they block.

In our approach a MAS is structured as a network of agents where the links between agents define a neighbourhood relationship. Agents in this network can execute joint actions with each other. We take a strong stance on speech act theory, by which all agent actions are illocutions which can be encapsulated as messages. In order to execute a joint action an agent a will search for a path through the network leading to a partner b where all the path's intermediate agents have granted a access to the next step in the path. Once the path is found, agents a and b can execute a joint action. In our approach the set of all actions agents can execute must be defined, and the normative behaviour defines which actions are permitted depending on the environment. Having to search for a path through the network in order to execute a joint action makes agents depend on other agents. Such dependence allows agents to block access to the network to those agents which violate the norms by executing actions which are forbidden. A violator agent is effectively ostracised from the network when enough agents have blocked it.

The peer norm enforcement techniques introduced in this chapter could be used in applications where the norms are well known to all the participants and whose compliance can be objectively tested. For instance, in a file sharing network where the file size cannot exceed 100MB, or in a news sharing application where certain linguistic expressions are forbidden. In order to join the file or news sharing systems, an agent would have to create at least one link to one of the agents already in the system thereby creating a social network of information sharing agents. Given that the file sharing or information sharing networks have a set of norms about the information that can be sent, sending information which does not fulfil these norms would be forbidden. By using the enforcement techniques proposed in the current chapter, those agents which violate the norm repeatedly would eventually be ostracised and could not continue harming the rest of the agents.

We have designed a totally distributed system that allows norm enforcement. Therefore, there are only two ways in which an agent can find out whether another agent is a norm violator: by being the partner in a joint action where the other agent executed a forbidden action, or by having the contents of a joint action, where a forbidden action was executed, being disclosed to it. In the MAS model we propose, we use encryption techniques that guarantee that the disclosure of the joint action contents can only be verified as truthful by the agents in the joint action path. This restriction removes any incentive to disclose non-existing joint actions, or to disclose to agents not in the joint action path.

The process through which an agent is ostracised is shown in Figure 4.2. At

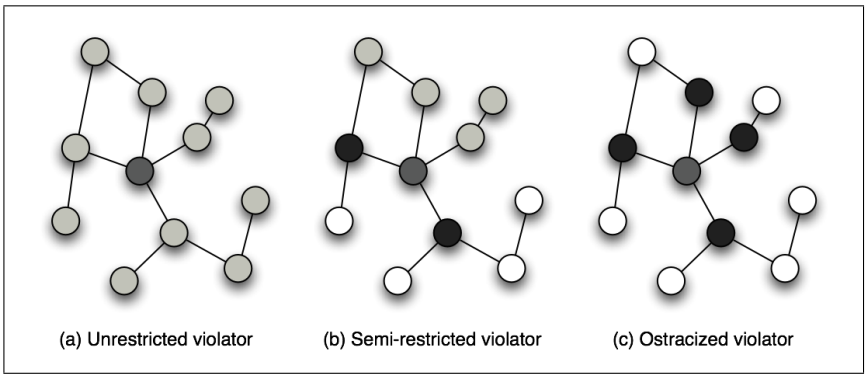

Figure 4.2: Ostracising a violator

first a norm violator (dark grey node) is believed to be an abiding member of the MAS, therefore the other agents will execute joint actions with it (the light grey nodes can execute joint actions with the violator). At some point the norm violator will execute a forbidden action and the partner of this joint action will realise this. Furthermore, the intermediate agents in the path between them will also know if the partner discloses the joint action contents. If agents knowing about this forbidden behaviour block the norm violator (black nodes are blocking the violator), its access to other agents in the network will be restricted (*i.e.,* white nodes cannot execute joint actions with the violator). When all its neighbours find out about its forbidden behaviour and block it, the norm violator is effectively ostracised.

When designing the multiagent system, our main concern was how effectively norm violators could be stopped, and what behavioural properties agents had to exhibit in order to speed up the ostracism process. Initial versions of this work, published in [Perreau de Pinninck et al., 2007, Perreau de Pinninck et al., 2008b], presented a model that has been improved in this chapter and from which we have extracted some analytical results. Furthermore, on our initial approaches we did some exploratory analysis that guided us towards formulating certain properties of the system that would account for a reduction on the number of norm violations. In this chapter we have generalised the model used in the above mentioned initial approaches by allowing more interactions than just those that can be modelled through game theory, thus, we have been able to concentrate on studying how the behavioural properties of the agents affect the abidance to the norms at the global level without dealing with the agent motivations. We have proven analytically that when all agents in the network exhibit certain behavioural properties, there exists an upper bound to the total number of illegal actions that can be executed. Nonetheless, since agents are autonomous, we cannot ensure that they will all exhibit such behavioural properties. In those cases where there is a subset of agents that apply the norms, we have shown that the number of norm violations executed against them also has an upper bound under certain conditions. Notwithstanding, there

are still cases in which either these condition do not hold or we want to continue to quantify norm violations to all agents. For these cases, we have run several experiments to support our claim that when agents exhibit such behavioural properties, the number of illegal actions that are executed is reduced. Furthermore, one of the analytical proofs supports our intuition that the network topology has an impact on the ostracism efficiency. The proof shows that the network size, *i.e.,* the total number of links in the graph, is the upper bound of illegal actions under certain conditions. We have also run experiments whose results support this hypothesis.

The remaining of the chapter is structured as follows. Section 4.2 describes the multiagent system model and Section 4.3 defines the agent behaviour model, some properties it can exhibit, and shows analytically how they influence norm enforcement. Section 4.4 presents a detailed description of the scenario employed in the experiments. Section 4.5 gives an account of the simulations, and analyses the resulting data. Section 4.6 provides some examples of how the model and techniques could be applied in real world applications. Finally, Section 4.7 presents a discussion on the chapter's results and future work that follows from this research.

4.2 The Model

The model described in this section defines a special kind of multiagent system (MAS) which is structured as a network with fixed links. We will refer to these special MAS as *multiagent networks* (MAN). Agents in a MAN may execute joint actions with others, in these joint actions only a finite set of actions can be executed by each agent (see Definition 4.2.4). We take a strong stance on speech act theory, by which all agent actions are illocutions.

Throughout the formalisation below the following types of symbols will be used: Latin capital letters refer to sets (*e.g.,* A). Lower case Latin letters refer to elements of sets (*e.g.,* $a \in A$). Lower case Greek letters refer to functions (*e.g.,* η). Finally teletype words are used to refer to concrete values (*e.g.,* `void`), and bold words for predicates (*e.g.,* **violator**). Furthermore, in all mathematical formula the variables are universally quantified unless specified otherwise.

The multiagent networks we define form a graph where the vertices are agents and the edges are direct communication channels between them. Two agents are neighbours if there is an edge between them. Furthermore, the model defines the set of actions that agents can execute, containing a special action (*i.e.,* `void`) that means that the agent refuses to interact.

Definition 4.2.1. A multiagent network is a tuple $\mathcal{N} = \langle A, \eta, C \rangle$ where:

- A is a finite set of *agents*.
- $\eta : A \to 2^A$ is a *neighbourhood function* returning an agent's neighbours such that it is:
  - irreflexive, *i.e.,* $\forall a \in A\ (a \notin \eta(a))$

  - undirected, *i.e.,* $\forall a, a' \in A\ (a' \in \eta(a) \leftrightarrow a \in \eta(a'))$

- C is a finite set of *actions* that agents can potentially execute, with a distinguished element `void`.

In this model, to be neighbours means to be linked through a direct communication channel. Nonetheless, if two agents in the network are not neighbours, they may still interact through a path in the network.

Definition 4.2.2. Given a multiagent network $\mathcal{N} = \langle A, \eta, C \rangle$, a *path* is a finite sequence of agents $p = \langle a_1, a_2, ..., a_n \rangle$ where $a_i \in A$, such that:

1. its length is greater than one, *i.e.,* $n > 1$;
2. any pair of consecutive agents are neighbours, *i.e.,* $a_{i+1} \in \eta(a_i)$ for $i = 1, ..., n-1$;
3. it contains no cycles, *i.e.,* $i \neq j$ iff $a_i \neq a_j$ for $i, j = 1, ..., n$.

Agent a_1 is referred to as the *initiator* and a_n the *partner*. The other agents in the path are referred to as *mediators*. Let P be the set of all paths in network $\mathcal{N}$. Let $\mu : P \to 2^A$ be the mediator function. Given a path $p \in P$, $\mu(p)$ is the set of mediators in a path, *i.e.,* $\mu(\langle a_1, a_2, ..., a_n \rangle) = \{a_i \mid 1 < i < n\}$.

A multiagent network defines how joint actions are executed and how agents observe them. During the execution of a MAN, two processes may occur: The joint action execution and the execution disclosure. The joint action execution process may be driven by the need of agents to act together. The disclosure process may be driven by the agents willing to make the contents of a joint action known in order to make norm violators identity known to others. This can either be motivated by revenge to the norm violator, or by altruism towards others that may encounter the same norm violating agent in the future. Nonetheless, we do not aim to study the motivations of agents.

The joint action execution process is made up of the following stages:

1. A path is *constructed* that links an initiator and a partner.
2. The initiator and partner agents execute a *joint action* through the constructed path.

Agents may be able to execute many joint actions in parallel, but we assume that all events they perceive can be ordered. The events an agent can perceive are either the proposal of neighbours as potential partners, the execution of actions by a partner in a joint action, or the disclosure of the contents of a joint action.

Definition 4.2.3. Given a multiagent network $\mathcal{N} = \langle A, \eta, C \rangle$ a *partner proposal* is a tuple $\langle a, a', A' \rangle$ such that:

- $a \in A$ is the agent seeking for a partner agent that queries for neighbours.

- $a' \in A$ is the agent that proposes a set of its neighbours as potential partners.
- $A' \subseteq \eta(a')$ is the set of potential partners proposed by a' which must be a subset of its neighbours

Let F be the set of all partner proposals in $\mathcal{N}$.

Definition 4.2.4. Given a multiagent network $\mathcal{N} = \langle A, \eta, C \rangle$ a *joint action* is a tuple $\langle p, c, d, J \rangle$ such that:

- $p = \langle a_1, a_2, ..., a_n \rangle$ is a path in P.
- $c \in C$ is the initiator's action (*i.e.,* a_1's action).
- $d \in C$ is the partner's action (*i.e.,* a_n's action).
- J is the set of previously executed joint actions either by the initiator or partner, and that have been observed by both (see discussion below Definition 5.1.3 about the actions observed by the agents from the environment). Therefore, $\langle p, c, d, J \rangle \notin J$

Let G be the set of all joint actions in $\mathcal{N}$. Consequently, $J \subseteq G$. A joint action $\langle p, c, d, J \rangle$ is of *mutual consent* when $c \neq \texttt{void}$ and $d \neq \texttt{void}$.

Definition 4.2.5. Given a multiagent network $\mathcal{N} = \langle A, \eta, C \rangle$, a *disclosure* is a tuple $\langle a, \langle \langle a_1, a_2, ..., a_n \rangle, c, d, J \rangle \rangle$ such that:

- $a \in A$ is the agent disclosing the joint action.
- $\langle \langle a_1, a_2, ..., a_n \rangle, c, d, J \rangle \in G$ is the joint action being disclosed.

The disclosing agent must be either the initiator or partner of the joint action, *i.e.,* $a = a_1$ or $a = a_n$. Let D be the set of all disclosures in $\mathcal{N}$.

A MAN has an associated environment containing the history of all events (*i.e.,* partner proposals, joint actions, and disclosures). The history is the only part of the *environment* that we model.[2] Agents have different perceptions of the environment, since they can only observe those events in which they are involved: being a seeker or a proposer in a partner proposal, being the initiator or partner of a joint action, or being a mediator of a disclosed joint action.

Definition 4.2.6. Given a multiagent network $\mathcal{N} = \langle A, \eta, C \rangle$, an environment is a tuple $e = \langle F', G', D' \rangle$ such that:

- $F' \subseteq F$ is a set of partner proposals.
- $G' \subseteq G$ is a set of joint actions.
- $D' \subseteq D$ is a set of disclosures.

[2] An environment, in general, could contain other pieces of information (*e.g.,* sensor readings) that we do not consider.

Let E be the set of all environments for $\mathcal{N}$.

A global environment would contain all events that occurred during the system execution. Although the global environment might not be stored in any place it is a useful mathematical construct. On the other hand, there is a local environment that each agent can observe. These local environments are partial views of the global environment, thus, the global environment is the union of all the agents' local environments. Furthermore, an agent is said to have observed a joint action if it is part of its local environment.

Definition 4.2.7. Given a multiagent network $\mathcal{N} = \langle A, \eta, C\rangle$, a *global environment* $e = \langle F', G', D'\rangle$, and an agent a, the *local environment* of agent a is $e|_a = \langle F'', G'', D''\rangle$, such that:

- $F'' = \{\langle a', a'', A'\rangle \in F' \mid a = a' \vee a = a'' \vee a \in A'\}$
- $G'' = \{\langle\langle a_1, a_2, ..., a_n\rangle, c, d, J\rangle \in G' \mid a = a_1 \vee a = a_n\}$
- $D'' = \{\langle a', \langle p, c, d, J\rangle\rangle \in D' \mid a \in \mu(p)\}$

Let $\theta : E \times A \to 2^G$ be the *observed joint action function*, where $\theta(e, a)$ is the set of all joint actions observed by agent a from the global environment $e = \langle F', G', D'\rangle$, *i.e.,* given $e|_a = \langle F'', G'', D''\rangle$ as defined above, $\theta(e, a) = G'' \cup \{g \in G' | \langle a', g\rangle \in D''\}$.

At the beginning of a MAN execution, the environment is always empty (*i.e.,* $e_0 = \langle\emptyset, \emptyset, \emptyset\rangle$), thus, no agent has observed any joint actions (*i.e.,* $\theta(e_0, a) = \emptyset$). Whenever a joint action is executed in an environment $e \in E$, the set J of previously executed joint actions is the intersection of the joint actions observed from the environment by the interacting agents up to the moment of execution, *i.e.,* $\forall\langle\langle a_1, a_2, ..., a_n\rangle, c, d, J\rangle \in G'$ $(J = \theta(e, a_1) \cap \theta(e, a_n))$.

The multiagent networks described in this chapter are normative. This means that an agent may be forbidden to execute some actions against another agent depending on the environment. In a MAN, the system designer defines a normative behaviour function that describes which actions an agent is permitted to execute in a joint action with another agent given the set of commonly observed joint actions. These commonly observed joint actions form part of the joint action so that any mediator can verify the partner agent's abidance to the normative behaviour if the joint action is disclosed.

Definition 4.2.8. A *normative behaviour* is defined as a function $\nu : 2^G \times A \times A \to 2^C$. Given the agents $a, a' \in A$, with the commonly observed joint actions $J \subseteq G$, $\nu(J, a, a')$ is the set of actions that a is permitted to execute in a joint action with a'. A joint action $g = \langle\langle a_1, a_2, ..., a_n\rangle, c, d, J\rangle$ is a *norm violation* when either $c \notin \nu(J, a_1, a_n)$ or $d \notin \nu(J, a_n, a_1)$. Our model allows agents to refuse to interact with norm violating agents. Therefore, it is always permitted to execute the `void` action against an agent that has executed a norm violation, *i.e.,* `void` $\in \nu(J, a, a')$ if there exists $\langle\langle a'_1, a'_2, ..., a'_n\rangle, c', d', J'\rangle \in J$ such that either $c' \notin \nu(J', a'_1, a'_n) \wedge a' = a'_1$ or $d' \notin \nu(J', a'_n, a'_1) \wedge a' = a'_n$ hold.

4.3 Behavioural Model

The previous section has introduced our model of a multiagent network. In this section we propose a behavioural model for the system, and we define some behavioural properties.

Executing joint actions and disclosure follow a specific algorithm in the current model which consists of three parts:

1. A path is *constructed* that links an initiator and a partner.
2. The initiator and partner agents execute a *joint action* through the constructed path.
3. Either the initiator or partner agent discloses the previously executed joint action.

4.3.1 Functional Model

In the presented behavioural model we define functions describing the behaviour among agents. We assume deterministic agents, thus, we may define functions that describe the system's behaviour. We cannot learn these functions from observations, since the only way to define these functions is to have access to the internals of all agents in the system. Nonetheless, we may have approximations of these behaviours that allow us to see whether they satisfy specific properties (see Section 4.3.5).

Definition 4.3.1. Given a multiagent network $\mathcal{N} = \langle A, \eta, C \rangle$, its *behaviour* is a tuple $\langle \alpha, \pi, \delta \rangle$, where:

- $\pi : A \times A \times E \to F \cup \{\perp\}$ is a *potential partners* function that models the agents' mediation behaviour. Given an initiator agent $a \in A$, a mediator agent $a' \in A$, and an environment $e \in E$, $\pi(a, a', e)$ is either a partner proposal from a' to a or the empty event.
- $\alpha : A \times A \times E \to G \cup \{\perp\}$ is an *action execution* function that models the agents' action execution behaviour. Given an initiator agent $a \in A$, a partner agent $a' \in A$, and an environment $e \in E$, $\alpha(a, a', e)$ is either the joint action executed by a and a' in the environment e or the empty event.
- $\delta : A \times A \times E \to D \cup \{\perp\}$ is a *disclosure* function that models the agents' disclosure behaviour. Given an initiator agent $a \in A$, a partner agent $a' \in A$, and an environment $e \in E$, $\delta(a, a', e)$ is either the disclosure uttered by any of the two participants (see Section 4.3.4) or the empty event. We assume that agents do not disclose joint actions more than once, *i.e.,* $\delta(a_1, a_n, \langle F', G', D' \rangle) \notin D'$.

Let B be the set of all system behaviours. The modelling of the system behaviour with functions is needed to prove the analytical results in Section 4.3.5. As a

notation abuse, we define an agent behaviour as the system behaviour when one of the agent input variables is fixed to a specific agent.

4.3.2 Constructing a path

In the first stage of the joint action execution process an initiator selects a path that leads to a partner with which to interact. Not all paths in the network fulfil the properties needed in order to be part of a joint action because they depend on the the network environment and the system behaviour.

Definition 4.3.2. Given a multiagent network $\mathcal{N} = \langle A, \eta, C \rangle$, an environment $e \in E$, and a behaviour $\langle \alpha, \pi, \delta \rangle \in B$, a path $p = \langle a_1, a_2, ..., a_n \rangle$ is said to be *socially feasible* when each agent a_{i+1} in the path is proposed as a potential partner to the initiator by the previous agent in the path (*i.e.,* a_i) ,*i.e.,* $\forall 1 < i \leq n$ $(\pi(a_1, a_i, e) = \langle a_1, a_i, A' \rangle \wedge a_{i+1} \in A')$.

During the path search process, the initiator agent will query agents for potential partners which are returned through partner proposal illocutions. The proposed partners must be a subset of the neighbours of the agent being queried. In order to construct the path, any graph search method may be used (see Section 4.4.2 for examples of search methods).

4.3.3 Executing a joint action

The second stage in the joint action execution process is to create the joint action. The joint action contains the feasible path that was constructed in the previous stage, the actions executed by the initiator and partner agents, and the set of joint actions observed by both of them from the environment. A joint action is executed because the initiator constructed a socially feasible path towards the partner. Nonetheless, both agents have the ability of executing the `void` action, which makes the joint action not of mutual consent.

By taking a strong stance on speech act theory, all actions are be executed through message passing. The messages containing the joint action are sent from one agent to another through the joint action path. In order for the selected actions to be private to the interacting agents the messages that contain the joint action are encrypted. This can be done by having a public key infrastructure (PKI) in which the public key of an agent is its identifier. Nonetheless, the use of a PKI involves some centralisation. This is why we choose to use a web-of-trust (WOT) approach implemented via OpenPGP through which we achieve the same results as using a PKI but without any centralisation. When executing a joint action, the message containing it would be encrypted using the recipient's public key. The encrypted messages would be passed along the joint action path, but none of the mediators should be able to decipher their content, thus keeping them private.

4.3.4 Disclosing joint actions

Disclosure is the process through which either the initiator or partner agents make the contents of a joint action observable to the mediator agents. In the previous section we have seen that the mediator agents have access to the encrypted message containing the joint action. This message has been encrypted using the public key of the destination agent, which is known to all. Therefore, if either the initiator or partner agent decide to disclose the contents of the executed joint action, they only need to send the decrypted contents to the mediators. The mediators can easily test whether the disclosed joint action contents are truthful by encrypting it using the public key of the recipient agent and verifying that the encrypted message matches the previous one that was sent through them. Since the original encrypted joint action is only known to the path mediators, only they can test its validity. Therefore, disclosure of joint actions is limited to the path's mediators. Furthermore, the mediators can test whether any of the actions was a norm violation by using the normative behaviour function ν.

The environment is updated as joint actions are executed and disclosed. In our MAN model the initiator and partner agents can disclose joint actions that have been previously executed by them only once. The environment is updated to include these illocutions whenever they are uttered.

4.3.5 Behavioural properties

In this section the properties of the proposed behavioural model are shown. They establish why a set of agents exhibiting certain behavioural properties enforce the norm by discouraging norm violations.

Definition 4.3.3. Given a multiagent network $\mathcal{N} = \langle A, \eta, C \rangle$ with a normative behaviour function ν, and given an environment $e \in E$, an agent a is a *norm violator* with respect to an agent a', noted as $\mathbf{violator}(a, a', e)$, if a executed a forbidden action in any of the joint actions observed by a'.

$$\begin{aligned}\mathbf{violator}(a, a', e) \leftrightarrow \quad & \exists \langle \langle a_1, a_2, ..., a_n \rangle, c, d, J \rangle \in \theta(e, a') \\ & ((a_1 = a \wedge c \notin \nu(J, a_1, a_n)) \vee (a_n = a \wedge d \notin \nu(J, a_n, a_1)))\end{aligned}$$

There are potentially many types of agent behaviours. We will discuss the properties of some of them: avoiding, blocking, protecting, and informing.

Definition 4.3.4. Given a multiagent network $\mathcal{N} = \langle A, \eta, C \rangle$, and a normative behaviour function ν, an action execution function α is said to be *violator avoiding* for an agent $a \in A$ when a executes the `void` action against norm violators, *i.e.,* $\mathbf{violator}(a', a, e) \wedge ((\alpha(a, a', e) = \langle p, \texttt{void}, d, J \rangle) \vee (\alpha(a', a, e) = \langle p', c', \texttt{void}, J' \rangle))$. Let the predicate $\mathbf{avoiding}(\alpha, a)$ hold when the action execution function α is violator avoiding for agent a. A behaviour is said to be violator avoiding for agent a if its action execution function is.

For the following proofs we define the function $\tau : 2^G \times A \times A \to 2^G$. Given a set of joint actions $G' \subseteq G$ and two agents $a, a' \in A$, $\tau(G', a, a')$ is the set of the given joint actions that were executed by the given agents, *i.e.,* $\tau(G', a_i, a_j) = \{\langle\langle a_1, a_2, ..., a_n\rangle, c, d, J\rangle \in G' \mid (a_1 = a_i \wedge a_n = a_j) \vee (a_1 = a_j \wedge a_n = a_i)\}$. We also define the function $\rho : 2^G \to 2^G$. Given a set of joint actions $G' \subseteq G$, and a normative behaviour function ν, $\rho(G')$ is the subset of the given joint actions which are norm violations and of mutual consent (see Definition 4.2.4), *i.e.,* $\rho(G') = \{\langle\langle a_1, a_2, ..., a_n\rangle, c, d, J\rangle \in G' | (c \notin \nu(J, a_1, a_n) \vee d \notin \nu(J, a_n, a_1)) \wedge c \neq \texttt{void} \wedge d \neq \texttt{void}\}$.

Lemma 4.3.5. *Given a multiagent network $\langle A, \eta, C\rangle$, and a normative behaviour function ν, with an environment $e \in E$, let $\langle \alpha, \pi, \delta\rangle \in B$ be the system's behaviour. If the action execution function is violator avoiding for agents a_i and a_j, then the number of norm violations in mutually consented joint actions by agents a_i and a_j is at most 1, i.e.,* $\mathbf{avoiding}(\alpha, a_i) \wedge \mathbf{avoiding}(\alpha, a_j) \to |\tau(\rho(\theta(e, a_i)), a_i, a_j)| \leq 1$.

Proof. We proceed by reductio ad absurdum. Let us assume that $\mathbf{avoiding}(\alpha, a_i) \wedge \mathbf{avoiding}(\alpha, a_j) \wedge |(\tau(\rho(\theta(e, a_i)), a_i, a_j)| > 1$. Let $g = \langle\langle a_1, a_2, ..., a_n\rangle, c, d, J\rangle$ be the joint action in $\tau(\rho(\theta(e, a_i)), a_i, a_j)$ executed the latest. This is known because by construction g contains all other joint actions in $\tau(\rho(\theta(e, a_i)), a_i, a_j)$ (see the discussion after Definition 5.1.3).

From the assumption we deduce that $\tau(\rho(\theta(e, a_i)), a_i, a_j) \setminus \{g\}$ cannot be empty. Since the action execution function is violator avoiding for agents a_i and a_j, and $\tau(\rho(\theta(e, a_i)), a_i, a_j) \setminus g \subseteq J$, thus containing at least one norm violation by agent a_i or a_j, then the joint action $\alpha(a_i, a_j, e')$, where e' is the environment at the moment of execution, should contain at least one `void` action. Therefore, g cannot be a joint action of mutual consent, *i.e.,* $g \notin \rho(\theta(e, a_i))$, which is a contradiction. Hence $\mathbf{avoiding}(\alpha, a_i) \wedge \mathbf{avoiding}(\alpha, a_j) \to |\tau(\rho(\theta(e, a_i)), a_i, a_j)| \leq 1$ as we wanted to prove. □

Theorem 4.3.6. *If the behaviour $\langle \alpha, \pi, \delta\rangle$ of a multiagent network $\mathcal{N} = \langle A, \eta, C\rangle$, with a normative behaviour function ν, contains an action execution function that is violator avoiding for all agents (i.e., $\forall a_i \in A$ ($\mathbf{avoiding}(\alpha, a_i)$)), then there exists an upper bound to the total number of potential norm violations in mutually consented joint actions that can be executed. This upper bound is $|A|(|A| - 1)/2$.*

Proof. The total number of norm violations in a multiagent network is equal to the sum of the norm violations happening between each pair of agents. Therefore, one can calculate the total number of norm violations by adding the potential norm violations between each pair of agents.

$$\frac{\sum_{a_i, a_j \in A, i \neq j} \tau(\rho(\theta(e, a_i)), a_i, a_j)}{2}$$

Given that all agents in the system have a violator avoiding behaviour, Lemma

4.3.5 applies to all pairs of agents. Therefore,

$$\frac{\sum_{a_i,a_j \in A, i \neq j} \tau(\rho(\theta(e,a_i)),a_i,a_j)}{2} \leq \frac{|A|(|A|-1)}{2}$$

which proves the theorem. □

The following result may be proved in much the same way as Lemma 4.3.5 and Theorem 4.3.6:

Corollary 4.3.7. *Given a multiagent network $\langle A, \eta, C\rangle$ with a normative behaviour function ν, and a behaviour $\langle \alpha, \pi, \delta\rangle$ that is violator avoiding for a group of agents $A' \subseteq A$ (i.e., $\forall a_i \in A'$ (**avoiding**(α, a_i))), then there exists an upper bound to the total number of potential norm violations in mutually consented joint actions that can be executed by an agent when the other agent in the joint action is part of A'. This upper bound is $|A'|(|A|-1)/2$.*

When the avoiding behavioural property is combined with other behavioural properties, the enforcement capabilities grow by reducing the number of potential norm violations.

Definition 4.3.8. Given a multiagent network $\langle A, \eta, C\rangle$ with a normative behaviour function ν, a potential partners function π is said to be *blocking* for agent $a \in A$ when the partner proposals it returns contain the empty set if the querier is a norm violator, *i.e.,* **violator**$(a_j, a, e)) \wedge \pi(a_j, a, e) = \langle a_j, a, A'\rangle \rightarrow A' = \emptyset$. Let the predicate **blocking**(π, a) hold when the function π is blocking for agent a. A behaviour is said to be blocking for agent a if its potential partners function is.

It can be easily shown that when the system's behaviour is avoiding and blocking for all agents, a smart norm violator would execute forbidden actions against agents in the network whose neighbours are accessible through some other path. Therefore, the upper bound to the number of potential norm violations would remain the same as if all agents had an avoiding behavioural property alone. Consequently, in order to lower the upper bound we explore other behavioural properties.

Definition 4.3.9. Given a multiagent network $\langle A, \eta, C\rangle$ with a normative behaviour function ν, a potential partners function π is said to be *protecting* for agent $a \in A$ when the partner proposals it returns never contain norm violators, *i.e.,* $\forall a_j \in \eta(a)($**violator**$(a_j, a, e)) \wedge \pi(a_k, a, e) = \langle a_k, a, A'\rangle \rightarrow a_j \notin A')$. Let the predicate **protecting**(π, a) hold when the function π is protecting for agent a. A behaviour is said to be protecting for agent a if its potential partners function is.

It is also straightforward to prove that when the system's behaviour is avoiding and protecting for all agents, a smart norm violator would execute forbidden actions against non-neighbouring agents first (*e.g.,* by using a depth first search algorithm). Therefore, the number of potential norm violations shall remain the

same as if all agents have an avoiding behavioural property alone. Even when all agents use a behaviour with avoiding, blocking, and protecting properties a depth first search for joint action partners to violate would maintain the upper bound. Therefore, other behavioural properties are needed to lower this upper bound.

Definition 4.3.10. Given a multiagent network $\langle A, \eta, C\rangle$ with a normative behaviour function ν, a disclosure function δ is said to be *informing* for agent $a \in A$ when it returns disclosures of norm violating joint actions executed against it, *i.e.,* $\delta(a, a', \langle F', G', D'\rangle) = \langle a, g\rangle \rightarrow (g = \langle\langle a, a_2, ..., a'\rangle, c, d, J\rangle \wedge d \notin \nu(J, a', a)) \vee (g = \langle\langle a', a_2, ..., a\rangle, c, d, J\rangle \wedge c \notin \nu(J, a', a))$. Let the predicate **informing**(δ) hold when the function δ is informing. A behaviour is said to be informing for agent a if its disclosure function is.

Finally, it is obvious that if all agents in the system have a norm violation with avoiding and informing properties, a smart norm violator would only execute forbidden actions against agents in a breadth first manner (*i.e.,* first through paths where it is a violator to all the mediators). In this way, the upper bound to the number potential norm violations remains the same as for a plain avoiding property.

Definition 4.3.11. A behaviour is full blocking for a given agent when it combines avoiding, blocking, protecting, and informing behavioural properties. Let the predicate **fullBlocking**$(\langle\alpha, \pi, \delta\rangle, a)$ hold when the behaviour $\langle\alpha, \pi, \delta\rangle$ is full blocking for agent a, *i.e.,* **fullBlocking**$(\langle\alpha, \pi, \delta\rangle, a) \equiv$ **avoiding**$(\alpha, a) \wedge$ **blocking**$(\pi, a) \wedge$ **protecting**$(\pi, a) \wedge$ **informing**(δ, a).

For the following proofs we define the function $\tau : 2^G \times A \times A \rightarrow 2^G$. Given two agents $a_i, a_j \in A$ and a set of joint actions $G' \subseteq G$, $\tau(G', a_i, a_j)$ is the set of joint actions executed by the first agent through a path where the second agent appears beside it, *i.e.,* $\tau(G', a_i, a_j) = \{\langle\langle a_1, a_2, ..., a_{n-1}, a_n\rangle, c, d, J\rangle \in G' | (a_1 = a_i \wedge a_2 = a_j) \vee (a_n = a_i \wedge a_{n-1} = a_j)\}$. We also define the function $\rho : 2^G \times A \rightarrow 2^G$. Given an agent $a \in A$, a set of joint actions $G' \subseteq G$, and a normative behaviour function ν, $\rho(G', a)$ is the subset of the given joint actions in which the given agent selected the norm violating action in mutually consented joint actions, *i.e.,* $\rho(G', a) = \{\langle\langle a_1, a_2, ..., a_n\rangle, c, d, J\rangle \in G' | ((c \notin \nu(J, a_1, a_n) \wedge a_1 = a) \vee (d \notin \nu(J, a_n, a_1) \wedge a_n = a)) \wedge c \neq \texttt{void} \wedge d \neq \texttt{void}\}$. Note that $\rho(\theta(e, a'), a)$ not being empty implies **violator**(a, a', e) but the opposite implication does not hold, since the **violator** predicate takes into account all joint actions with norm violations whereas ρ only takes into account those joint actions that are of mutual consent.

Lemma 4.3.12. *Let $\mathcal{N} = \langle A, \eta, C\rangle$ be a multiagent network with a normative behaviour function ν, a behaviour $b = \langle\alpha, \pi, \delta\rangle$, and an environment $e \in E$, where a_j and a_k are two neighbouring agents, i.e., $a_j \in \eta(a_k)$. If the behaviour is full blocking for each agent $a_i \in A$, i.e., $\forall a_i \in A$ (**fullBlocking**(b, a_i)), then the number of mutually consented joint actions executed by a_j in which it executed a norm violation through a path where a_k appears beside it, is at most 1, i.e.,* **fullBlocking**$(b, a_i) \rightarrow |\tau(\rho(\theta(e, a_j), a_j), a_j, a_k)| \leq 1$.

Proof. We proceed by reductio ad absurdum. Let us assume that $\forall a_i \in A(\textbf{fullBlocking}(b, a_i)) \wedge |(\tau(\rho(\theta(e, a_j), a_j), a_j, a_i)| > 1)$. Let $g = \langle\langle a_1, a_2, ..., a_{n-1}, a_n\rangle, c, d, J\rangle$ be the joint action in $\tau(\rho(\theta(e, a_j), a_j), a_j, a_i)$ executed de latest. This is known because by construction g contains all other joint actions in $\tau(\rho(\theta(e, a_j)), a_j, a_i)$ (see the discussion after Definition 5.1.3). From Definition 4.3.2, the paths of all executed joint actions must be feasible given the environment at the time of execution.

If $n = 2$ then we follow a similar reasoning as that of Lemma 4.3.5. Given that $\tau(\rho(\theta(e, a_j), a_j), a_j, a_i) \setminus \{g\}$ is not empty and α is avoiding for agent a_j, a_j's action would have been `void` and the executed joint action not of mutual consent, which brings about a contradiction.

Otherwise, when $n > 2$ there are two options to consider: i) paths of the form $\langle a_j, a_i, ..., a_n\rangle$ or ii) paths of the form $\langle a_1, ..., a_i, a_j\rangle$. In both cases from the assumption it follows that $\tau(\rho(\theta(e, a_j), a_j), a_j, a_i) \setminus \{g\} \neq \emptyset$. Furthermore, since the disclosure function is informing for all agent in the network, then all agents in the path of a norm violation have observed the joint action, *i.e.,* $\forall\langle\langle a_1, a_2, ..., a_n\rangle, c, d, J\rangle \in \rho(\theta(e, a_j), a_j), \forall k \in 1, ..., n\ (\langle\langle a_1, a_2, ..., a_n\rangle, c, d, J\rangle \in \theta(e, a_k))$. Therefore, $\tau(\rho(\theta(e', a_k), a_j), a_j, a_i) \neq \emptyset$, where e' is the environment right before the execution of g.

Since the observed potential partners function is blocking and protecting for all agents, the path in the former option would not be feasible because $\pi(a_j, a_i, e')$ would contain an empty partner set, since a_j is a violator with respect to a_i's observed joint actions and π is blocking for agent a_i. In the latter option the path would not be feasible because $\pi(a_1, a_i, e')$ would not contain a_j since it is a violator with respect to a_i's observed joint actions and π is protecting for agent a_i.

All of the possible cases bring about a contradiction. Hence, $\forall a_i \in A(\textbf{fullBlocking}(b, a_i) \rightarrow |\tau(\rho(\theta(e, a_j), a_j), a_j, a_k)| \leq 1$ as we wanted to prove. □

Theorem 4.3.13. *In a multiagent network $\mathcal{N} = \langle A, \eta, C\rangle$ with a normative behaviour function ν, and a behaviour $b = \langle\alpha, \pi, \delta\rangle$ which is full blocking for all agents (i.e., $\forall a_i \in A$ (**fullBlocking**(b, a_i))), there exists an upper bound to the total number of potential norm violations in mutually consented joint actions that can be executed. This upper bound is twice the number of links of the network, i.e., $\sum_{a_i \in A} |\eta(a_i)|$.*

Proof. From Lemma 4.3.12 follows that if all agents have a full blocking behaviour, one single agent a is able to violate the norm at most once for each neighbour it has (*i.e.,* $|\eta(a)|$). Therefore, the total norm violations will be less than the sum of all agent neighbours (*i.e.,* $\sum_{a \in A} |\eta(a)|$) which is twice the number of links in the network. □

In multiagent networks where the average neighbours per agent is less than half the population, it pays for agents to exhibit a full blocking behaviour, as opposed to a simpler avoiding behaviour, since less norm violations are possible. On the other hand, for densely connected networks an avoiding behaviour is

sufficient. For such networks disclosing and storing all the norm violations is useless.

Interestingly, Theorem 4.3.13 implies that the structure of the multiagent network has an impact on norm enforcement. More densely connected networks are more prone to norm violations. On a single agent scale, agents with more neighbours can get away with more norm violations. Furthermore, with more neighbours comes a higher risk of becoming a victim of norm violations.

Nonetheless, as mentioned earlier in an open system it is highly probable that not all agents will exhibit the same behavioural properties. The following corollary generalises the results of full blocking behaviours when just a subset of agents exhibit them.

Corollary 4.3.14. *Given a multiagent network $\mathcal{N} = \langle A, \eta, C \rangle$ with a normative behaviour function ν, a behaviour $b = \langle \alpha, \pi, \delta \rangle$, a subset of agents $A' \subseteq A$ that form a connected component, and a function $\phi : A \times 2^A \to 2^P$, where $\phi(a, A')$ is the subset of elements in A' reachable from a through a path of agents not in A'. If the behaviour is full blocking for all agents in this connected component (i.e., $\forall a_i \in A$ ($\mathbf{fullBlocking}(b, a_i)$)), then there exists an upper bound to the total number of potential norm violations in mutually consented joint actions that can be executed by one of the agents in the joint action when the other agent is part of A'. This upper bound is $\sum_{a \in A'} |(\eta(a) \cap A') \cup \phi(a, A')| + \sum_{a \notin A'} |\phi(a, A')|$.*

Proof. It is easily seen that agents in the enforcing component A' will only be able to execute norm violations against the other agents in A' once through each neighbour in A', and if it has neighbours outside A' it will be able to execute norm violations against other agents in A' as many times as agents in A' it can reach through a path of agents outside A'. Whereas, those agents outside A' will only be able to execute a norm violation against agents in A' as many times as agents in A' it can reach through a path of agents outside A'. □

4.4 The Scenario

Section 4.3 showed analytical results for multiagent networks where all agents exhibit the same type of enforcement behaviour. In a system where agents are assumed to be autonomous, one cannot expect that all agents will exhibit the same behaviour. Even though some of the analytical results have been generalised for subsets of the network that did exhibit the same enforcement behaviour (see Corollaries 4.3.7 and 4.3.14), we would like to verify how different types of behaviours would work in plural societies. Another issue with the analytical model is that it assumes that each executed joint action is atomic, meaning that no change in the environment can occur through the process of finding a socially feasible path. This may not be a valid assumption in real scenarios where the path may be found to be feasible, but stops being so before the joint action is eventually executed through the path. This may happen when an agent receives disclosed contents of a joint action that is a norm violation after the creation of a path leading to it or coming from it, but before the joint

action is executed. For the reasons above, we have run experiments that test the following hypotheses in a simulated environment without the restrictions of uniform societies and atomic execution:

Hypothesis 1. *Stricter enforcement behaviours reduce the number of norm violations.*

Hypothesis 2. *A larger ratio of agents with enforcing behaviours reduces the number of norm violations.*

Hypothesis 3. *The network topology influences the number of norm violations.*

The scenario that has been used in the experiments follows the model described in the previous sections. The simulated environment consists of a multi-agent network where agents take turns to start an interaction by first searching for a feasible path, then executing a joint action through it, and finally going through a disclosure stage in which they may or may not send the joint action contents to the path mediators.

4.4.1 Agents

As seen in Section 4.2, the system has a behaviour which is modelled via three functions. There is a potentially infinite number of possible behaviours. Nonetheless, we do not aim to cover all the different behaviours, only a reduced set of coarse grained behaviours to test how the enforcement techniques work against norm violators.

The agents in our experiments can be classified under one of the three following types: meek, violator, and enforcer. Each type having a behaviour with different properties. Some of these properties were already defined in Section 4.3.5, such as violator avoiding, blocking, protecting, and informing. The rest are defined below:

Definition 4.4.1. Given a multi-agent network $\mathcal{N} = \langle A, \eta, C \rangle$ with a normative behaviour function ν, and a behaviour $\langle \alpha, \pi, \delta \rangle$, we define the following behavioural properties for an agent $a \in A$:

- *Disclosing*: Will always report all its neighbours as potential partners, *i.e.,* $\pi(a_i, a, e) = \langle a_i, a, A' \rangle \rightarrow A' = \eta(a)$.

- *Friendly*: Always consents to act jointly, *i.e.,* $(\alpha(a, a_i, e) = \langle \langle a, a_2, ..., a_i \rangle, c, d, J \rangle \rightarrow c \neq \texttt{void}) \wedge (\alpha(a_i, a, e) = \langle \langle a_i, a_2, ..., a \rangle, c, d, J \rangle \rightarrow d \neq \texttt{void})$.

- *Abiding*: Always abides by the norm, *i.e.,* $(\alpha(a, a_i, e) = \langle \langle a, a_2, ..., a_i \rangle, c, d, J \rangle \rightarrow c \in \nu(J, a, a_i)) \wedge (\alpha(a_i, a, e) = \langle \langle a_i, a_2, ..., a \rangle, c, d, J \rangle \rightarrow d \in \nu(J, a, a_i))$.

- *Hiding*: Will avoid joint actions with those that disclose norm violations, *i.e.,* $(\alpha(a, a_i, \langle F', G', D' \rangle) = \langle \langle a, a_2, ..., a_i \rangle, c, d, J \rangle \wedge \exists \langle a_i, \langle \langle a'_1, ..., a, ..., a'_n \rangle, c', d', J' \rangle \rangle \in D' \ ((a'_1 = a_i \wedge d' \notin \nu(J', a'_n, a'_1)) \vee (a'_n = a_i \wedge c' \notin \nu(J', a'_1, a'_n))) \rightarrow c = \texttt{void}) \vee (\alpha(a_i, a, \langle F', G', D' \rangle) = \langle \langle a_i, a_2, ..., a \rangle, c, d, J \rangle \wedge \exists \langle a_i, \langle \langle a'_1, ..., a, ..., a'_n \rangle, c', d', J' \rangle \rangle \in D' \ ((a'_1 = a_i \wedge d' \notin \nu(J', a'_n, a'_1)) \vee (a'_n = a_i \wedge c' \notin \nu(J', a'_1, a'_n))) \rightarrow d = \texttt{void})$

- *Discreet*: Never discloses the joint action contents to mediators, *i.e.,* $\delta(a, a_i, e) = (\langle a_i, g \rangle \vee \bot) \wedge \delta(a_i, a, e) = (\langle a_i, g \rangle \vee \bot)$.

The agent type depends on which of the previous behavioural properties hold. As mentioned earlier the three main categories of agents we consider are: meeks, violators, and enforcers. A behaviour $\langle \alpha, \pi, \delta \rangle$ is of type *meek* for a given agent if it is disclosing, friendly, abiding, and discreet. On the other hand a behaviour is of type *enforcer* for a given agent if it is avoiding and abiding. Furthermore, an enforcer behaviour may also be informing, blocking, or protecting (all of which are enforcement properties). An enforcement behaviour b_e is stricter than another $b_{e'}$ if b_e satisfies all the enforcement properties that $b_{e'}$ does and more. Finally, a behaviour is of type *violator* if it is disclosing, friendly, violating, and discreet. Violator agents in our simulations try to maximise the number of times they violate the norm. Furthermore, violator agents may also exhibit a hiding behaviour and will avoid selecting an enforcer agent as partner.

One can envision other more fine-grained behavioural functions for the different types of agents, such as enforcers that forgive violators by removing blockages after a certain number of rounds following the norm violation. The number of rounds after which forgiveness is granted could be made dependant on the number of known norm violations, making for an even more fine-grained enforcement behaviour. One could also think of more sophisticated violator behaviours, such as violators that block enforcers or violators that do not violate against agents that have exhibited blocking or protecting behaviours. Finally, all these behaviours could be probabilistic (*e.g.,* if implemented to depend on the reputation of the other agent). Nonetheless, we have chosen not to study of these variations, as our aim is to show that certain enforcement techniques reduce the number of norm violations for some applications and set a limit to the amount of norm violations possible in certain scenarios, rather than to find the optimal enforcement behaviour for any application.

4.4.2 Variables

In order to execute the simulations we had to give values to the different variables that define our experimental scenario. Table 4.1 shows the list of variables considered, the range of values they may take, and the values we have used for the experiments. Some of these variables are specifically mentioned in the hypotheses. Hypothesis 2 makes a reference to the ratio of enforcing agents. The higher the value of this variable, the smaller the number of norm violations. Hypothesis 1 has an implicit reference to the three variables on the type of enforcement being used. Simulations where one of these variables is set to

Name	Type
Agents	$2, \infty$
Rounds	$1, \infty$
Topology	Small world, Scale Free, Tree, Ring, Random,...
Violator	$[0, 1]$
- Enforcer avoiding	$\bot, \top$
- Partner Search	Random, BFS, DFS,...
Enforcer	$[0, 1]$
- Blocking	$\bot, \top$
- Protecting	$\bot, \top$
- Informing	$\bot, \top$

Table 4.1: Simulation variables

true have fewer norm violations than simulations where they were set to false. Finally, Hypothesis 3 references the network topology explicitly by expecting some topologies to allow different numbers of norm violations.

It is quite straightforward to see that the number of rounds and the ratio of violating agents will also have an impact in the number of norm violations. Not so obvious is the relation between the type of violator behaviour (both by choosing whether or not to avoid enforcers, and in the way they search for feasible paths), or the agent order which dictates their position in the network and the order in which they attempt to interact. Furthermore, one can presume that the total number of agents will not have an impact in the number of norm violations. Nonetheless, the experiments have not been designed to test any of these latter hypotheses.

For the variables taking natural numbers as values we have selected four points from a pseudo-logarithmic scale. For the variables representing ratios we have selected three points in a linear scale. In our simulations, agents cannot both enforce and violate the norm. Therefore, the enforcing agent ratio creates a constraint for the violating agent ratio and vice-versa (*e.g.,* a MAN with 80% of enforcers can have at most 20% of violators). In order to bypass this constraint, the selected values are always below 50%. Furthermore, 0% has not been selected as a value, since societies without enforcers or violators are not relevant to test our hypotheses. For variables that do not have a numerical range we have selected those values that we deem interesting: the network topologies that have been chosen are sufficiently different among each other, and most can be found in real societies, and partner search strategies that have been chosen are those that had been considered in Section 4.3.5, and which can maximise the number of norm violations done by agents (Breadth First Search and Depth First Search) plus a fully random search (norm abiding agents will always follow a random search strategy). Finally, the agent order has been chosen at random, although having a finite set of permutations of agents as possible orders.

The topologies we have selected are defined below. A *tree* is a connected

undirected graph without cycles. Trees tend to have a large diameter and an average clustering coefficient[3] of 0. Trees are abstractions of highly hierarchical structures. A *random* graph is one that has been generated through some random process. Random graphs have a small diameter and a low average clustering coefficient. A *small-world* graph is one which has a small diameter but an average clustering coefficient orders of magnitude higher than those of a random graph with the same order and size. Finally, a *scale-free* graph is one where the number of neighbours follows a power-law distribution. These types of graphs have small diameter and low average clustering coefficient. They are neither as structured as trees nor as unstructured as random graphs. Networks presenting the small-world property or the free-scale property can be found in many social systems.

4.4.3 Feasible path search algorithm

Algorithm 1 is executed by the initiator agent in order to find a socially feasible path leading to a partner. First, if the current mediator is different than the initiator and it is selected as partner, then the algorithm returns the path together with the visited agent set passed as parameters to the function (line 5). Otherwise the initiator queries the current mediator for the set of potential partners (line 1) and the function iterates through the set of non-visited potential partners until either a feasible path is found or the set is exhausted. If the set is exhausted but no feasible path has been found, then the algorithm returns an empty path together with an updated set of visited agents (line 17). Otherwise it returns the path from the recursive call (line 19).

Algorithm 1 defines an abstraction of the search for socially feasible paths. Nonetheless, agents can implement different search strategies, as explained previously (random, breadth-first, or depth-first). These strategies are implemented through particular definitions of the functions `get_element_from` and `select_as_partner`. The implementation of these functions for the breadth-first and depth-first searches is straightforward and will not be discussed. The random search implements the `get_element_from` function by returning an agent in a purely random fashion, and it implements the `select_as_partner` by returning true with a certain probability. In our experiments this probability has been set to 0.3, which constructs paths with length following a geometric distribution with mean 10/3. This is enough to guarantee that all agents can interact with one another.

Many different methods exist to generate the previously mentioned graphs. We have chosen the following: The generated tree structures are k-ary trees ($k = 9$), *i.e.,* a tree with up to k children per node. Random graphs have been generated following the Erdös-Rényi model [Erdös and Rényi, 1960], in which the probability that an edge between two vertices exists is 0.1. Finally, the small world graphs have been generated following the Watts-Strogatz

[3] A graph's average clustering coefficient is the probability that any two neighbours of a given agent are also neighbours.

Algorithm 1: Feasible path search - $\texttt{path}(a_1, a_i, V, p)$

Input: $a_1 \in A$ - initiator agent
Input: $a_i \in A$ - current mediator
Input: $V \in 2^A$ - visited agents
Input: $p \in P$ - feasible path
Output: feasible path
Output: set of visited agents

request a_i the potential partners for a_1;
$\langle a_1, a_i, A' \rangle \leftarrow \pi(a_1, a_i, e)$;
$N \leftarrow A' \setminus V$;
if $a_i \neq a_1 \wedge$ $\texttt{select\_as\_partner}(a_i, N)$ **then**
    return (p, V);
else
    while $N \neq \emptyset \wedge \neg stop$ **do**
        $a_j \leftarrow \texttt{get\_element\_from}(N)$;
        $N \leftarrow N \setminus \{a_j\}$;
        $V \leftarrow V \cup \{a_j\}$;
        $(p', V) \leftarrow \texttt{path}(a_1, a_j, V, p \cdot a_j)$;
        if $p' \neq \langle\rangle$ **then**
            $stop \leftarrow \top$;
        end
    end
    if $N = \emptyset \wedge \neg stop$ **then**
        return $(\langle\rangle, V)$;
    else
        return (p', V);
    end
end

model [Watts and Strogatz, 1998] starting with a ring lattice of degree 10, and a rewiring probability of 0.02. Finally, scale-free graphs have been generated using the Barabasi-Albert model [Barabasi and Albert, 1999] with a size and order similar to that of the small-world and random graphs[4].

The experiment consists of exhaustive simulations with all the possible combinations of variable values. The metric we have used to test the hypothesis is the norm violation rate, *i.e.,* the ratio of forbidden actions executed per violator agent and round. This metric is a normalisation of the total forbidden actions so that we can compare simulations with different number of agents and rounds.

4.5 Simulations

This section shows the results of the experiments that have been executed following the scenario specified in Section 4.4. For each of the hypotheses presented we explain the statistical analysis that has been realised on the experimental data and the results of such analysis.

In order to test the three hypotheses we have executed a factorial experiment. The experiment consists of running a number of simulations (20 in total) for each of the parameter combinations in Table 4.2.[5] Before executing the statistical significance test we verified that the resulting data had a normal distribution through a Quantile-Quantile test, which is a precondition for certain statistical tests.

In order to test if there was a significant relationship between the independent variables, *i.e.,* the parameters in the simulation, and the dependent variable, *i.e.,* percentage of the total potential interactions per violator agent that resulted in a norm violation (from now on we will refer to this as the norm violation rate), we ran an analysis of variance (ANOVA) with the experimental data. The results of the ANOVA test demonstrate that all the independent variables were statistically significant with a p-value under 0.001. This information alone serves to support Hypothesis 3, since the topology variable proved to be significant. Data showing which variable values brought about lower norm violation rates is needed in order to support the other two hypotheses.

In order to verify which variable values did better for each of the variables that interested us with respect to the hypotheses, we ran post-hoc comparisons using Tukey's test (see the results in Table 4.3). One test indicated that higher percentages of enforcers implied a smaller mean of norm violation rates, thus supporting Hypothesis 2. Furthermore, another Tukey test indicated that the best network topology in reducing the norm violation rate was the random topology. Close behind were the small world and scale free topologies, and, lagging behind, the tree topology was the one to do worst. Finally, Tukey tests on the enforcement behaviours showed that the three tested behaviours helped to reduce the norm violation rate. This information supports Hypothesis 1,

[4] All networks are generated randomly using one of the previous methods, and the agents are placed randomly in a graph position.

[5] There are $27,648$ total combinations. In total $552,960$ simulations were executed.

Name	**Values**
Agents	$10, 20, 100, 200$
Rounds	$10, 20, 100, 200$
Topology	Small world, Scale Free, Tree, Random
Violator	$0.1, 0.3, 0.5$
- Enforcer avoiding	$\bot, \top$
- Partner Search	Random, BFS, DFS,...
Enforcer	$0.1, 0.3, 0.5$
- Blocking	$\bot, \top$
- Protecting	$\bot, \top$
- Informing	$\bot, \top$
Name	**Type**
Joint Actions	$\mathbb{N}$
Norm Violations	$\mathbb{N}$
Integrity Constraints	**Formula**
Partitioning	Violator + Enforcer $<= 1.0$

Table 4.2: Simulation variables

since stricter enforcement behaviours reduced the norm violation rate. Out of the three behavioural properties, the property that produced the best results was the protecting property which lowered the norm violation rate by an average of 7.37%. Next came the blocking property which lowered the norm violation rate by an average of 3.05%, and last was the informing property that barely lowered the norm violation rate, a meagre 0.26%.

The Tukey test results on different topologies were a surprise to us as they countered our intuitions. Experiments on initial approaches[Perreau de Pinninck et al., 2007, Perreau de Pinninck et al., 2008b] had showed that tree networks were more efficient in reducing norm violation rates than other topologies. This, together with the analytical results obtained in Theorem 4.3.13, made us think that the tree network would still be the one to achieve the best enforcement performance. Nonetheless, the results from the experiments showed that it is random networks that have the best enforcement performance and tree networks have the worst. Such results are brought about by the fact that trees have the highest betweenness centrality. Agents in a high betweenness centrality position will collect more joint action contents through disclosure. In our initial approaches, only enforcer agents benefited from their high betweenness centrality, which made tree networks more efficient for enforcement. In the current scenario, violator agents also collect information to find out who the enforcers are in order to avoid them. Therefore, they also benefit from high betweenness centrality positions, which makes trees worse for enforcement. Furthermore, our simulations restricted the maximum percentage of enforcers

Variable	Values	Difference	Lower bound	Upper bound
Enforcers	10% - 30%	8.83%	8.71%	8.95%
Enforcers	30% - 50%	8.96%	8.84%	9.08
Enforcers	10% - 50%	17.79%	17.67%	17.91%
Topology	TR - RM	13.37%	13.22%	13.52%
Topology	TR - SW	13.01%	12.86%	13.16%
Topology	TR - SF	12.73%	12.58%	12.88%
Topology	SF - RM	0.64%	0.49%	0.79%
Topology	SW - RM	0.36%	0.21%	0.51%
Topology	SF - SW	0.28%	0.13%	0.43%
Protecting	false - true	7.37%	7.29%	7.45%
Blocking	false - true	3.05%	2.97%	3.13%
Informing	false - true	0.26%	0.18%	0.34%

Table 4.3: Tukey test results

to 50%, making it easy for violators to find non-enforcers with which to execute norm violating actions.

Another result from the Tukey test that surprised us was the low reduction brought about by the informing behavioural property. At first glance, we thought that disclosing contents of norm violations would be a good enforcement behaviour. Nonetheless, the slight improvement brought about by the informing behavioural property made us look into this in more detail. Further tests showed that in simulations were the number of rounds is larger than the number of agents, the informing behavioural property actually increases the mean norm violation rate by 0.40% on average. Furthermore, when the number of rounds is equal to the number of agents there is no significant difference between results with and without the informing behavioural property. Finally, when the number of rounds is smaller than the number of agents the norm violation rate is decreased by 1.15%. Table 4.4 shows the results of the Tukey tests with the data subsets mentioned before. This is due to the fact that violators, in our simulations, avoid enforcers and these are detected when they disclose joint action contents. It is during bootstraping (when the number of rounds is small and agents had no time to interact with one another) that disclosure allows enforcers to quickly discover all violators. As the simulation continues, it is the violators who end up discovering all enforcers through their informing behavioural property, and can thus easily avoid them. This tendency could be changed if those enforcers disclosing the joint action contents could select a subset of the mediator agents to which the contents is sent. This way enforcers would avoid being discovered by violators and the enforcement benefits of disclosure be improved.

Data set	Value pair	Diff.	Lower b.	Upper b.
Rounds > Agents	$\perp - \top$	-0.40%	-0.54%	-0.26%
Rounds = Agents	$\perp - \top$	-0.09%	-0.26%	0.08%
Rounds < Agents	$\perp - \top$	1.15%	1.04%	1.26%

Table 4.4: Informing variable Tukey test results for subsets of the simulation data.

4.6 Applications

This section presents two applications that could make use of our model and enforcement techniques. In order to apply these techniques, the applications need to have an objective norm function which is shared by all, the norm violating behaviours should not be too sophisticated, and norm violations should only affect those agents involved in the joint action.

The first application is a distributed forum for information sharing in which a group of agents forms a network of contacts through which they exchange information. In this forum there are norms specifying the valid information contents that can be shared. These norms must be verifiable objectively by any agent. Furthermore, agents violating these norms would either do it continuously in the same way as spammer agents in fora, or it would violate occasionally by mistake or because the recipient does not care for the specific norm violation. Both types of violators fit the specification of Section 4.4

The second application is for a network of hardware nodes (such as a sensor network) which co-operate by executing a specific protocol in order to manipulate information. The norm in this case would define the protocol to be followed. Such protocol ought to be objectively verifiable. Furthermore, all nodes are assumed to work well unless there is a hardware malfunction, in which case they will start breaking the protocol indefinitely.

In the following subsections we will show how our model and enforcement techniques are applied to the applications, and the results that can be achieved from these enforcement techniques.

4.6.1 Information sharing forum

In order to model the distributed information sharing forum application, we must start by defining the MAN $\mathcal{N} = \langle A, \eta, C \rangle$ with a normative behaviour function ν. Having A be the set of participating agents in the forum (let us assume it is 100 in total), η be the function that describes the connections among the agents (let it be a network with 500 links and the average degree is 10), C is the set of all content that can be shared, and ν is the function that specifies which contents are valid.

A joint action in this application means that one agent sends some information through a path of agents in the network to another agent. The action executed by the receiving agent indicates what it did with the information.

When it executes the `void` action, it is telling the initiator that the information transfer has been immediately aborted. It is as if it had never been received. A norm in this application might specify that the maximum size of exchanged information is 10MB, and that specific file types such as mp3 or text documents containing words from a previously established list of insults are not allowed. This norm predicate is objectively verifiable and is not influenced by the environment. Therefore, we will not mention the environment in the rest of this application. Furthermore, the `void` action is always permitted.

Let us assume that there are ten agents that are norm violators. Five of which are full spammers which will always send content that is not allowed by the norms. The other five will send invalid content rarely when they think they can get away with it. The results from Section 4.3.5 tell us that: if all agents in the application follow an avoidance behavioural property, the maximum number of norm violating contents that can be sent is 9900; if all agents exhibit a full blocking behaviour, this number is reduced to 1000. Nonetheless, since there are only 10 violator agents, when all agents exhibit an avoidance behavioural property the upper bound is reduced to 990 (see Lemma 4.3.5) and when they all exhibit a full blocking behaviour it is reduced to 100 (see Lemma 4.3.12).

In case that the norm violators do not exhibit enforcement behaviours, the premises from the theorems would not hold but those of the corollaries would and we would apply their conclusions. Assuming that all norm abiding agents apply an avoidance behavioural property, then the maximum number of norm violating contents shared with norm abiding agents would be 900, and approximately 90 in case they all exhibited full blocking behaviours. This has been calculated assuming that all agents (including norm violators) have ten neighbours and probabilistically 9 of these neighbours will be norm abiding agents, and that all norm abiding agents form a connected component in the network.

When the agent's behaviours are not organised in any of the previous ways, the results from the simulations can still tell us how the system would behave. The application designer would then be interested in providing a free and accessible implementation of the information sharing agent that would embed a full blocking behaviour and leave other implementations to the users, thus, creating a cost for the implementation of other behaviours. Furthermore, when possible, the designer would be interested in organising the network randomly, since it was the random network that got the best enforcement performance.

4.6.2 Self-repair system

The self repairing network application is defined through the model as $\mathcal{N} = \langle A, \eta, C \rangle$, where A is the set of all nodes (let us assume a total of 100), η describes the connections among nodes, C defines the set of interaction data that can be sent, and a normative behaviour function ν that specifies the protocol for interaction.

In this application a joint action means that the initiator agent sends some data to the partner agents as part of a broader protocol. The protocol defined by the function ν is a simple query answer protocol where the `void` action can only

be executed by the partner agent as long as there has been a protocol violation in the past. When the partner agent executes the `void` action it is letting the initiator know that it is not listening to its message. In this application the environment consists of the previous joint actions executed by the agents in the joint action.

In this application the system designer has complete control over the agents' behaviours and the network topology. The only reason for a violator to exist is due to hardware malfunction. Therefore, once an agent violates a norm, chances are that it will continue to misbehave until its hardware is repaired. Furthermore, the malfunctioning node has no capabilities to evade enforcement through sophisticated norm violation behaviours.

From the description of the application we deduce that it is in the best interest of the application designer to make all nodes have a full-blocking behaviour. Since agents that have failed will not have a hiding behavioural property to evade enforcement, a tree network might seem the most appropriate topology. Nonetheless, since a node that has failed will probably stop reporting its neighbours correctly (which is the same as if the agent would block everyone in the network), then a tree network is dangerous as it would easily compromise a whole sub-tree. Therefore, the best topology would be a random network with an average degree ensuring that the network forms a connected component.

When the self repairing application is designed as explained above, the upper bound in the number of failed protocols would be the number of links in the network. This number is equal to the number of agents times the tolerance to errors per agent. When the probability that edges exist in a random graph is $\frac{2 \ln n}{n}$, where n is the number of vertices, the probability that the network is connected tends to 1 [Erdös and Rényi, 1960]. Therefore, to ensure that a random network is connected, the number of links would be $\ln n(n-1)$. Since our application comprises 100 agents, then the number of links would be at least 456. In order to make sure that errors in hardware do not split the network, the system designer should allow for more links (*e.g.,* 500). Finally, in the previous setup the maximum number of protocol failures (*i.e.,* norm violations) allowed would be 500, or an average of 5 per agent.

4.7 Discussion

We have provided a model for a multiagent system structured as a network where agents interact under a defined normative behaviour. Under this model, a set of enforcement techniques have been proposed to reduce the number of norm violations, namely: avoidance, blocking, protecting, and informing. Via analytical means we have shown that, when all agents (or a subset of them) exhibit behaviours with enforcement properties, the number of executed norm violations has an upper bound. Nonetheless, agents are autonomous and may decide to save up resources by not applying sanctions. In order to apply sanctions, agents must remember which other agents have executed a norm violating action in the past and take this information into account when helping to

build socially feasible paths. If there are agents in the system exhibiting this free-riding behaviour, the analytical results in Section 4.3 do not always hold. However, the results of the experiments in Sections 4.4 and 4.5 show us how such a system could be expected to behave. Firstly, the higher the number of agents using enforcement techniques the smaller the number of times violator agents would be able to execute norm violating actions. Secondly, using more of the enforcement techniques implies less norm violations. Nonetheless, we encountered results that challenged our intuitions. The experiments showed that disclosure of joint action contents does not always reduce norm violations, since overuse of disclosure can put violators under notice and help them avoid future enforcement. Therefore, disclosure should be restricted to trusted agents or be used only at bootstrap. Finally, the network structure is an important factor in reducing norm violations. Notwithstanding, we had expected tree networks to continue to be the best at reducing norm violations, as in the initial approaches, and this did not happen. When violators use disclosure information to spot enforcers they also benefit from the high centrality positions possible in tree networks, making tree networks the worst topology for enforcement for sophisticated violators that manage to "climb the ladder". The dependence network formalism could be a good lead to study the power agents have to enforce norms.

Builders of an application, in which the enforcement techniques in this chapter could be used, should try to get as many agents in the system to enforce the norm through the four techniques provided: avoiding, blocking, protecting, and informing. We have not aimed to study what could motivate a selfish agent to act in specific ways. We have provided information about which behaviours benefit the society by reducing the number of norm violations. It would be interesting to study how to create incentives so that agents exhibit the enforcement behaviours in order to diminish the potential incentives for executing norm violations. Furthermore, if builders have total control over the network topology, they should create a tree in which all the non-leaves would be under his control and would use all the full blocking behaviour. Otherwise, the tree network should be avoided. The impact of other network parameters (*e.g.*, clustering factor, diameter, number of links per agent, number of paths between agents, and position of agents) on norm enforcement should be studied, in order to help individual agents decide how to influence the network topology for its benefit. The study of the position of agents may prove specifically interesting for this purpose. For instance, if an agent is a cut vertex in the graph and it uses the protecting behavioural property against its neighbours it can break the graph in two, not allowing abiding agents to interact. Furthermore, a violator that is in such a position may threaten to split the graph in order to avoid enforcement. These problems can be circumvented with new links.

This chapter does not treat network dynamics by which the network topology changes by adding and removing links, or by adding new agents to the system. Even so, agents should be extremely weary when adding new links in order not to allow agents to execute more norm violations through them. In Chapter 5 we have taken network dynamics into account, in which case agents can be

sanctioned when they are too promiscuous in adding new links through which unsatisfactory interactions take place.

The enforcement techniques in this chapter are meant for scenarios in which the agents do not know a priori with which agent to share information. Notwithstanding, they could also be applied to scenarios in which the partner agent is known and a feasible path in the network is to be found. In such a case routing mechanisms used in networking could be used in order to send the requests through the social networks. Since the partner agent would be known from the start of the process, the protecting behaviour would have to be modified but others could also be added.

Some applications that would benefit from the techniques in this chapter have been provided in Section 5.10. These applications have to fulfil the restrictions imposed by the model, such as the objectivity of the norm and bareness of the agent behaviours available. Nonetheless, other applications that do not fulfil these restrictions could also benefit from the concepts defined here. In order to tackle these, Chapter 5 studies the impact of subjective norms, where each agent decides what is satisfactory for it, and how more sophisticated enforcement behaviours could achieve better enforcement performance against more sophisticated violating behaviours.

Chapter 5

Ostracism under Uncertainty

Ostracism has been shown to be a useful tool in enforcing behaviour. Chapter 4 presented some enforcement techniques through which agents that did not behave as expected could be ostracised. Nonetheless, the work in that chapter was developed under assumptions that significantly restricted their applicability. Those assumptions are:

- The network topology is static.
- There is no defined mechanism through which new agents can enter.
- Contact relationships are undirected.
- Expected behaviour is global and known to all agents.
- Adversarial behaviour is simple.

The mechanisms described in Chapter 4 cannot be implemented in most widespread large-scale distributed systems (*e.g.,* filesharing applications, VoIP and messaging applications, and GRID), since the previous assumptions do not hold. This chapter defines a new model for a multiagent network (MAN) that can be applied to those distributed systems in which the previous assumptions do not hold.

The main differing characteristics of the model in this chapter are: i) Contact relationships are directional. The new model defines the contact relationship as an explicit trust relationship. Therefore, the relationship is directed and not necessarily reciprocal, *e.g.,* the fact Albert trusts Brenda does not imply that Brenda trusts Albert back. ii) A mechanism is provided through which contact relationships can be added and removed. Through this mechanism new agents can join the system and its network topology is allowed to change through time. iii) Each agent has a subjective definition of what a satisfactory behaviour is.

This definition is not necessarily shared or known to others, but it can be modelled via the feedback given by the interacting agents about an interaction that has taken place. Therefore, having Claude complain about Dorian does not imply that Edward will complain about Dorian too. Nonetheless, by modelling the complaining behaviour, Frank can estimate the probability with which Dorian will be complained about by Edward (*i.e.,* reputation mechanisms are applicable). iv) Agents, specially those trying to cheat the system, exhibit complex behaviours. Therefore, simple enforcement mechanisms may not always work, since the agents for which they are intended may exhibit behaviours that can subvert these mechanisms (*e.g.,* collusion, badmouthing, ballot-stuffing, and others). The algorithms used to model the behaviours of other agents ought to be robust to these attacks. Furthermore, even agents that are not trying to cheat the system may be complained about. Therefore, the enforcement mechanism must allow for forgiveness.

A well-known paradigm for distributed applications for which these new assumptions hold, is peer-to-peer (P2P) networks. P2P networks are composed of many participants that share their resources with others in the network without the need of a centralised co-ordination mechanism. A computer user joins a P2P network in order to interact with other computer users through the use of client software that connects to other client software in the same overlay network. The assumptions in the model defined in this chapter fit nicely to P2P applications: the network topology changes, peers can enter and leave the network, there is no pre-defined expected behaviour, the definition of a satisfactory interaction is subjective, and adversaries exhibit complex behaviours (*e.g.,* virus spread, Sybil attacks). The only new addition that is not normally present in P2P networks is that of directed links. Notwithstanding, undirected links can be simulated with directed links. Given the suitability of P2P as a potential application for the model to be described in this chapter, in what follows the terms agent and peer will be used indistinctly, and *peer* refers to both the computer user and the P2P client software.

The aim of the work in this chapter is to provide enforcement techniques that allow agents to maximise their subjective satisfaction. This subjective measure is not publicly known, only what can be observed is known. Therefore, to be more precise, the chapter studies how the enforcement techniques increase the amount of *positive feedback* for the interactions between agents. The blocking technique shown in Chapter 4 is coupled with different reputation mechanisms in order to see how they fare against malicious peers that try to subvert the system through different types of attacks: badmouthing, *i.e.,* sending misleading feedback about prior interactions; ballot-stuffing, *i.e.,* sending feedback of non-existing interactions; milking or dynamic personality, *i.e.,* getting good assessments by satisfying other peers in order to cheat later on; collusion, *i.e.,* forming a collective with other peers to try to subvert the assessment system; Sybil, *i.e.,* creating many false identities that collude; and finally, whitewashing, *i.e.,* changing the peer's identification in order to avoid negative assessments from previous feedback.

Scalability is an important issue addressed in P2P applications (not so much in MAS). Therefore, the tests take into account the amount of messages sent by each reputation mechanism. Efficacy is important when enforcing behaviour, but it may come at the cost of efficiency. When the number of agents is low, it may not be a problem. But for large-scale systems efficiency is key, otherwise the original purpose of the system can be hampered.

The interaction protocol in this chapter is different to that presented in Chapter 4. The main difference being that the interaction partner is chosen before a path is found between the two (see Section 5.2 for a more detailed definition of the interaction protocol).

We use a Friend-to-Friend (F2F) network structure in which each peer is connected to a set of peers which it "knows", *i.e.,* its *contacts*. Each peer creates its own certificate and signs the certificates of its contacts, thus making a web of trust (WOT). By running a WOT, authentication and privacy can be guaranteed without the need for a centralised Public Key Infrastructure (PKI). Furthermore, the interaction requests would be routed through a path of contacts towards the target peer. In a P2P application any peer can connect through the internet to any other peer. This differs from the work in Chapter 4 in which agents could only connect to their neighbours. Nonetheless, having a WOT gives agents the choice to discard requests that have not been routed through a contact. This allows agents to force others to route their request through the MAN, thus becoming an enforcement technique. When this technique is applied, in order to interact a peer needs to be the contact of at least one other peer already in the network.

Feedback about interactions can also be sent to any peer in the network. Nonetheless, peers can choose to discard feedback of interactions for which they did not route the request. This feedback gathering strategy brings about the following outcomes: smaller message overhead, smaller impact of ballot-stuffing attacks, and less feedback available to each peer. In order to address the last issue, the blocking enforcement technique is integrated into the routing mechanism. Since the neighbours of a peer are bound to have more feedback regarding it, they can perform the best assessments. Peers decide whether to block requests given the probability that the initiator and target will be unsatisfied with the interaction.

Blocking and avoiding enforcement techniques are used in conjunction with a trust assessment algorithm. In this work the most significant algorithms have been made available to the peers. One of these algorithms (Peertrust [Xiong et al., 2004]) has been modified in order to harness the information available in the request routes. Furthermore, new scalable algorithms have been defined in order to be robust against the different adversarial attacks by harnessing the request route information of previous interactions. In order to test the enforcement techniques in conjunction with the different trust assessment algorithms, we have run experiments that compare them. These experiments have shown that the modification to Peertrust and the new algorithm have a message overhead orders of magnitude lower.

The rest of the chapter is organised as follows: Section 5.1 defines the multiagent network and the contents of the different messages that are sent as part of the interaction protocol. Section 5.2 gives an in depth description of the interaction protocol by using an example. Section 5.3 provides the definition of the behavioural model of the agents, and gives analytical properties of some agent behaviours. Section 5.4 provides a description of the different types of fraud and how encryption, signature, and reputation techniques can be used to avoid them. Section 5.5 compares the different trust assessment mechanisms in the literature that have been implemented. Section 5.6 describes the new trust assessment mechanisms that we have developed. All of these trust assessment mechanisms have been compared analytically in Section 5.7, and through the experiments defined in Section 5.9.

Section 5.8 describes the decisions made by the agents during the interaction process. Section 5.10 gives two examples of real applications to which the model and enforcement techniques in this chapter can be applied. Finally, Section 5.11 discusses the results of the model and the experiments.

5.1 The Model

This section describes the mathematical model for our multiagent system. This was already done in Chapter 4. Nonetheless, we will modify it here to take into account the differences studied in this chapter. These differences stem from the changes in the assumptions. Firstly, there is no global norm defined by an external authority. Secondly, the neighbourhood relation is now directed. Thirdly, there is no partner search since the intended partner is known at the beginning of the interaction protocol. Therefore, the protocol changes and so do the events and their contents. Fourthly, the interaction is not mediated, only the interaction bootstrap. Finally, there is no static network as the agents and their neighbourhood links can change at any time.

In the work presented here, there is a protocol for the interaction between agents. We call this the interaction protocol because its purpose is getting agents to interact. The first part of the protocol is the interaction bootstrap. An agent sends an interaction request, which gets routed through the network towards its intended target. If the request reaches the target, the target can accept to interact by acknowledging the request. This ends the bootstrap part of the protocol. After the acknowledgement reaches the protocol initiator, the joint action between the initiator and target begins through a private channel. After the joint action is executed the last phase of the protocol starts, the feedback stage. In this stage, each of the partners can send information about the joint action to the agents in the network.

Agents may be able to interact in parallel, but we assume that all events they perceive can be ordered. Given that we take a strong stance on speech act theory, these events can be encapsulated in illocutions. All illocutions have the same structure: they are composed of the sender identifier, the receiver identifier, and the content. The content is either an identity certification, a

certification cancellation, an interaction request, or an interaction feedback. From now on we refer to an illocution containing a specific content, as a specific content illocution, *e.g.,* an illocution containing an interaction request is an interaction request illocution. We assume that agents discard those illocutions whose recipient identifier does not match their own.

Definition 5.1.1. Let A be the set of all agent identifiers. An *illocution* is a tuple $\langle a, a', ct \rangle$ where $a \in A$ is the identifier of the illocution sender, $a' \in A$ is the identifier of the illocution receiver, and ct is the illocution contents, whose structure depends on the contents being sent. Let $\mathbb{I}$ be the set of all illocutions.

As in the previous chapter, a MAN has an associated environment containing the history of all illocutions. The environment could contain other information, but the history of illocutions is the only part of the environment that we model. In this chapter the illocution contents are different than in the previous chapter. Therefore, the structure of an environment also changes to incorporate the events in this model. Agents have subjective and partial perceptions of the environment, since they can only observe those illocutions that where either sent or received by them.

Definition 5.1.2. Given the set of illocutions $\mathbb{I}$, an *environment* is any subset of illocutions $e \subseteq \mathbb{I}$. Let $E = 2^{\mathbb{I}}$ be the set of all environments.

The environment contains all illocutions that have been uttered during the execution of the system. Such environment does not need to be stored in any place, but it is a useful mathematical construct. On the other hand, an agent could easily store a local view of the environment, containing those illocutions that where either sent or received by it.

Definition 5.1.3. Given a set of agent identifiers A, an environment e, and an agent with identifier $a \in A$, the *local environment* of agent a in e is $e|_a = \{\langle a_s, a_r, ct \rangle \in e \mid a_s = a \vee a_r = a\}$

The initial environment of a MAN is always empty (*i.e.,* $e_0 = \emptyset$). Furthermore, upon entering the MAN, the local environment of an agent will also be empty. This carries an underlying assumption: two agents can never possess the same agent identifier, either simultaneously or at different times. In real systems this assumption may not always hold but using an appropriate identification scheme (as the one described in Section 5.4) we can guarantee this assumption.

5.1.1 Illocution Content

The content of illocutions can be of different types, depending on the part of the interaction protocol that is taking place. In this section we define the different content types: identity certification, certification cancellation, request, and feedback.

The identity certification serves to acknowledge the identity of an agent by another agent. The sender agent utters an identity certification illocution when

it trusts the certified agent to be who it says it is. This mechanism follows a web of trust (WOT) approach, which is the mechanism we use for authentication. On the other hand, the certification cancellation states that an agent cancels its previous endorsement of another identity. An agent identifier becomes active when another agent certifies it, and stops being so when all the agents that had previously certified it cancel their certifications. The interaction protocol is defined so that only agents with an active identifier can interact with others. Making this the first of our enforcement mechanisms.

Definition 5.1.4. Given a set of content identifiers I, and a set of agent identifiers A, an *identity certification* is a tuple $\langle i, a, a' \rangle$ where $i \in I$ is the request identifier, $a \in A$ is the identifier of the certifying agent, and $a' \in A$ is the identifier of the agent being certified. Let C be the set of all identity certifications. We define the function $\zeta : E \to 2^C$ to represent the set of agent certifications in a given environment, *i.e.,* given an environment $e \in E$, $\zeta(e) = \{c \in C \mid \exists \langle a_r, a_s, c \rangle \in e\}$.

A *certification cancellation* is a tuple $\langle i, a, a' \rangle$ where $i \in I$ is the content identifier, $a \in A$ is the identifier of the agent removing its certification, and $a' \in A$ is the identifier of the agent whose identity certification is being cancelled. Let D be the set of all certification cancellations. We define the function $\chi : E \to 2^D$ to represent the set of certification cancellations in a given environment, *i.e.,* given an environment $e \in E$, $\chi(e) = \{d \in D \mid \exists \langle a_r, a_s, d \rangle \in e\}$.

Given an environment $e \in E$, $A_e \subseteq A$ is the set of *active agent identifiers*, *i.e.,* $A_e = \{a' \in A | \exists \langle i, a, a' \rangle \in \zeta(e) \ \wedge \neg \exists \langle i, a, a' \rangle \in \chi(e)\}$.

An agent with an active identifier (from now on we will use "agent" to refer to "agent with an active identifier") that wants to interact with another agent must get an interaction request delivered to the agent with which it wants to interact. Since enforcing agents will only listen to interaction requests in illocutions uttered by agents whose identity they have certified, most requests will have to be delivered through a path of agents that is made explicit in the request.

Definition 5.1.5. Given a set of agent identifiers A, and a set of content identifiers I, an *interaction request* is a tuple $\langle j, a_i, a_t, p \rangle$ such that:

- $j \in I$ is the content identifier.
- $a_i \in A$ is the identifier of the *initiator* agent, *i.e.,* the agent that created the original interaction request.
- $a_t \in A$ is the identifier of the *target* agent, *i.e.,* the agent for which the interaction request is intended.
- p is the route followed so far by the interaction request (see Definition 5.1.6).

Let R be the set of all interaction requests. An interaction request $\langle j, a_i, a_t, p \rangle$ is said to *entail* another $\langle j', a'_i, a'_t, p' \rangle$ when they share the same content identifier $j = j'$, initiator identifier $a_i = a'_i$, target identifier $a_t = a'_t$, and the route p

entails the route p' (see Definition 5.1.6). Let the predicate **entails** $\subseteq R \times R$ define the entailing relationship between two requests, where the first request entails the second. Let the function $\rho : E \to 2^R$ represent the set of interaction requests in a given environment, *i.e.,* given an environment $e \in E$, $\rho(e) = \{r \in R \mid \exists \langle a_r, a_s, r\rangle \in e\}$.

An interaction request contains information about the route the request has travelled in order to help the recipient of the request illocution decide what to do with the request. When the target agent is also the last agent in the request route recipient list, we say that the request has *reached its target*, in which case the target agent must decide whether or not it wants to interact with the initiator agent. Otherwise, when the recipient agent is not the target, it will decide whether it forwards, blocks, re-routes, or discards the request. A request is discarded if the recipient agent does not send an entailing request illocution. A request is forwarded by sending an entailing request illocution to an agent not already in the request route. A request is re-routed or blocked by sending an entailing request illocution to the agent that sent the request illocution. When blocking, the request route informs that the agent has blocked the request. Whereas, by re-routing the agent sending the entailing request says that it wants to forward the request but cannot because it has no agent to forward it to.

The request route contains information about the decisions taken by all the agents in the path that the interaction request has travelled.

Definition 5.1.6. Given a set of agent identifiers A, a *request route* is a tuple $\langle r, b, rr\rangle$ where $r \in A^*$ is a sequence of agent identifiers, and $b, rr \subseteq A$ are sets of agent identifiers. The sequence r contains all recipient agents, *i.e.,* those agents that have received a request entailed by the original. The set b contains all blocking agents. Finally, the set rr contains all re-routing agents. Therefore, all agents in r that are not in either b or rr are those that have effectively forwarded the request to the target. The order in the recipient sequence describes the order in which the agents formed part of the request delivery. Request routes $\langle\langle a_1, a_2, ..., a_n\rangle, b, rr\rangle$ must satisfy the following rules:

- All identifiers in the blocking and re-routing sequence must appear in the recipient sequence, *i.e.,* $\forall a \in b\ (\exists l\ (1 \leq l \leq n \wedge a = a_l)) \wedge \forall a \in rr\ (\exists l\ (1 \leq l \leq n \wedge a = a_l)$.
- No duplicates allowed in the recipient sequence, *i.e.,* $\forall j, k\ (1 \leq j < k \leq n \to a_j \neq a_k)$.
- An identifier in the re-routing sequence cannot appear in the blocking sequence and viceversa, *i.e.,* $b \cap rr = \emptyset$

Let P be the set of all request routes. A request route $\langle\langle a_1, a_2, ..., a_n\rangle, b, rr\rangle$ is said to entail another $\langle\langle a'_1, a'_2, ..., a'_{n'}\rangle, b', rr'\rangle$, when its recipient sequence is the beginning of the recipient sequence in the request route it entails, and the blocking and re-routing sets are includes in the sets of the request route it entails, *i.e.,* $n < n' \wedge \forall j\ (1 \leq j \leq n \to a_j = a'_j) \wedge b \subseteq b' \wedge rr \subseteq rr'$.

Those interaction request illocutions that are sent to the appropriate agent, and with a properly constructed request route (see Definition 5.1.7) are said to be correct for the agent receiving them. Those interaction request illocutions that are not correct are discarded by the receiving agent. For example, an agent a receiving a request illocution where the sender identity has not been certified by it will discard the request. Making this another of our enforcement mechanisms.

Definition 5.1.7. Given an agent with identifier $a \in A$, and an environment $e \in E$, an interaction request $r = \langle j, a_i, a_t, \langle\langle a_1, a_2, ..., a_n\rangle, b, rr\rangle\rangle$ as content of the illocution $\langle a_s, a_r, r\rangle$ is *correct* for agent a if the following properties hold:

- The recipient is a, *i.e.,* $a_r = a$.
- Either the identity of a_s is certified by a or the identity of a is certified by a_s, r is being blocked or re-routed by a_s, and r is entailed by a request that a sent in an illocution to a_s, *i.e.,* $a_s \in A_{e|_a} \vee (a \in A_{e|_{a_s}} \wedge \exists r' \in \rho(e|_a)\ (\mathbf{entails}(r', r) \wedge (a_s \in b \cup rr))$.
- Either a is the last element of the recipient list (*i.e.,* $a = a_n$) or all agents in the recipient sequence after a have blocked or re-routed the request and there exists an interaction request illocution in a's environment with a as the sender that entails r, *i.e.,* $\exists k\ (1 \leq k \leq n \wedge a_k = a \wedge \forall l\ (k < l \leq n \rightarrow a_l \in b \cup rr)) \wedge \exists \langle a'_s, a'_r, r'\rangle \in e|_a\ (r' \in \rho(e|_a) \wedge \mathbf{entails}(r', r) \wedge a'_s = a)$.

Let the predicate $\mathbf{correct} \subseteq \mathbb{I} \times A \times E$ hold when the given interaction request illocution is correct for the given agent identifier in the given environment.

When an agent receives a correct interaction request for which it is the target, it decides whether or not it wants to interact with the initiator agent. If it does, it will send an interaction request illocution where the contained interaction request is the one it received. We call these special interaction requests acknowledgements. Upon receiving an acknowledgement illocution an agent knows that the target of a interaction request accepts to interact with its initiator. Nonetheless, an agent can only verify that an acknowledgement is justified if it previously sent an entailing interaction request illocution.

Definition 5.1.8. Given a set of agent identifiers A, and a set of content identifiers I, an *acknowledgement* illocution is an interaction request illocution $\langle a_s, a_r, \langle j, a_i, a_t, \langle\langle a_1, a_2, ..., a_n\rangle, b, rr\rangle\rangle\rangle$ where the illocution sender is the request target and the request has reached its target, *i.e.,* $a_s = a_t = a_n$. Let K be the set of all acknowledgement illocutions.

Given an agent with identifier $a \in A$, and an environment $e \in E$, an acknowledgement $\langle a_s, a_r, \langle j, a_i, a_t, \langle\langle a_1, a_2, ..., a_n\rangle, b, rr\rangle\rangle\rangle$ is said to be *justified* for the receiving agent a if the following properties hold:

- The agent receiving the illocution is its recipient, *i.e.,* $a = a_r$.

- There exists a correct request sent by agent a that entails the acknowledged request, *i.e.,* $\exists\langle a, a'_r, r'\rangle \in e|_a$ $(r' \in \rho(e|_a) \wedge \textbf{correct}(\langle a, a'_r, r'\rangle, a, e) \wedge \textbf{entails}(r', \langle j, a_i, a_t, \langle\langle a_1, a_2, ..., a_n\rangle, b, rr\rangle\rangle))$.

Let the predicate **justified** $\subseteq \mathbb{I} \times A \times E$ hold when the given illocution is a justified acknowledgement for the given agent identifier, with the given environment. We define the function $\alpha : E \to 2^K$ to represent the set of acknowledgements in a given environment, *i.e.,* given an environment $e \in E$, $\alpha(e) = e \cap K$.

When the interaction request initiator receives a justified acknowledgement associated to a request it sent, a direct communication channel between the initiator and target is created and they execute a joint action through this channel. When an interaction follows the process described throughout this section (*i.e.,* first a request is sent by the initiator; this initial request entails other correct requests, one of which eventually reaches the target; then the target sends a justified acknowledgement to the initiator; finally, an interaction between both occurs), then the joint action is said to be consensual.

Definition 5.1.9. Given a set of agent identifiers A, and a set of request identifiers I, an *interaction* is a tuple $\langle j, a_i, a_t\rangle$ where $j \in I$ is the identifier of the request that triggered the interaction, $a_i \in A$ is the identifier of the request's initiator, and $a_t \in A$ is the identifier of the request's target. Let $\mathcal{I}$ be the set of all interactions.

Given an environment $e \in E$, an interaction is said to be *consensual* when there exists an acknowledgement in $\alpha(e)$ that is justified for the interacting initiator, with an acknowledged request where the interacting agents are the initiator and partner. Let the predicate **consensual** $\subseteq \mathcal{I} \times E$ hold when the given interaction is consensual. Mathematically speaking: $\textbf{consensual}(\langle j, a, a'\rangle, e) \leftrightarrow \exists\langle a_s, a_r, \langle j, a_i, a_t, p\rangle\rangle \in \alpha(e)$ $(\textbf{justified}(\langle a_s, a_r, \langle j, a_i, a_t, p\rangle\rangle, a, e) \wedge a_i = a \wedge a' = a_t)$.

The contents of consensual interactions are completely private, as opposed to the joint actions modelled in Chapter 4, where the joint action contents travelled encrypted through a path of mediators. Since there are no global norms governing the interaction contents between agents, the mediator agents cannot verify the compliance of the interaction even if they can access its contents. Notwithstanding, there is some subjective measure of the interactions that are appropriate, *i.e.,* all agents are able to assess whether an interaction they participated in is satisfactory to them. Agents send feedback to express the degree of satisfaction achieved from an interaction. An agent can send any number of feedback messages as long as they conform to a specific structure. Nonetheless, feedback is legitimate only when it refers to a consensual interaction. An agent can verify that a feedback is legitimate even if its local environment does not contain the acknowledgement, since the request is the same in the acknowledgement as in the feedback, the agent can check that the request in the feedback is entailed by a request it sent.

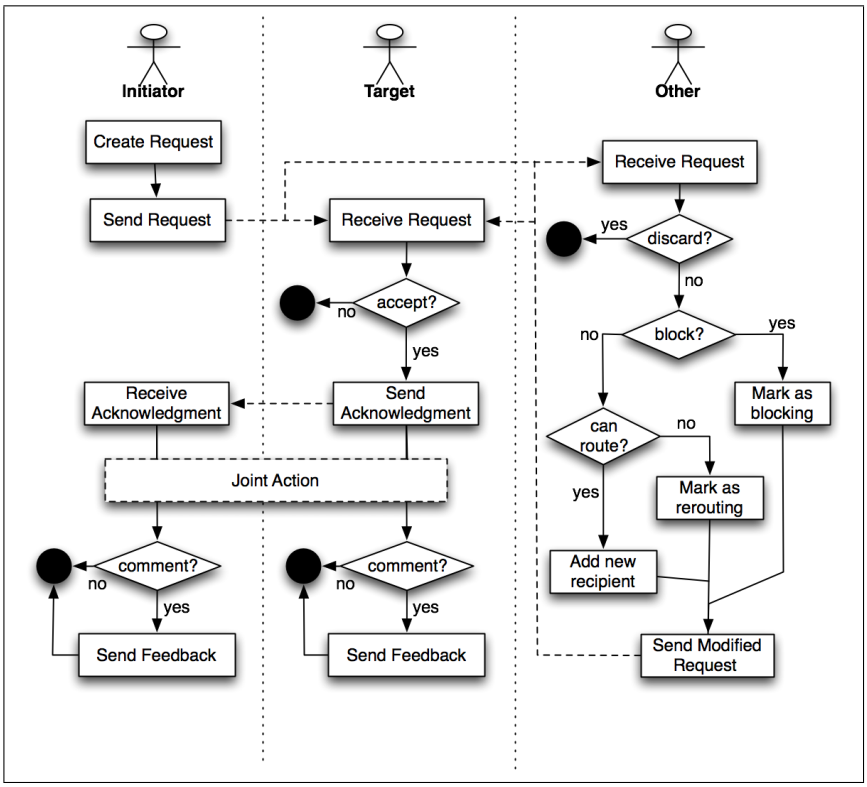

Figure 5.1: Interaction protocol

Definition 5.1.10. Given a set of agent identifiers A, and a set of content identifiers I, a *feedback* is a tuple $\langle\langle j, a_i, a_t, p\rangle, a_f, v\rangle$ where $\langle j, a_i, a_t, p\rangle \in R$ is an interaction request that has reached its target, $a_f \in A$ is the identifier of the agent giving the feedback, and $v \in [0, 1]$ is a real number that gives a value to the satisfaction. The agent giving the feedback must be either the request's initiator or target, *i.e.,* $a_i = a_f \vee a_t = a_f$. Let F be the set of all feedback.

Given an agent with identifier $a \in A$, and an environment $e \in E$, a feedback $\langle r, a_f, v\rangle \in F$ is said to be *legitimate* when there exists an interaction request illocution in the environment sent by a that is correct for a and entails the feedback request, *i.e.,* $\exists\langle a, a'_r, r'\rangle \in e|_a\ (r' \in R \wedge \mathbf{correct}(\langle a, a'_r, r'\rangle, a, e) \wedge \mathbf{entails}(r', r))$. Let the predicate $\mathbf{legitimate} \subseteq \mathbb{I} \times A \times E$ hold when the given illocution contains a legitimate feedback for the given agent in the given environment.

We define the function $\phi : E \to 2^F$ to represent the feedback in a given environment, *i.e.,* given an environment $e \in E$, $\phi(e) = \{f \in F \mid \langle a_s, a_r, f\rangle \in e\}$.

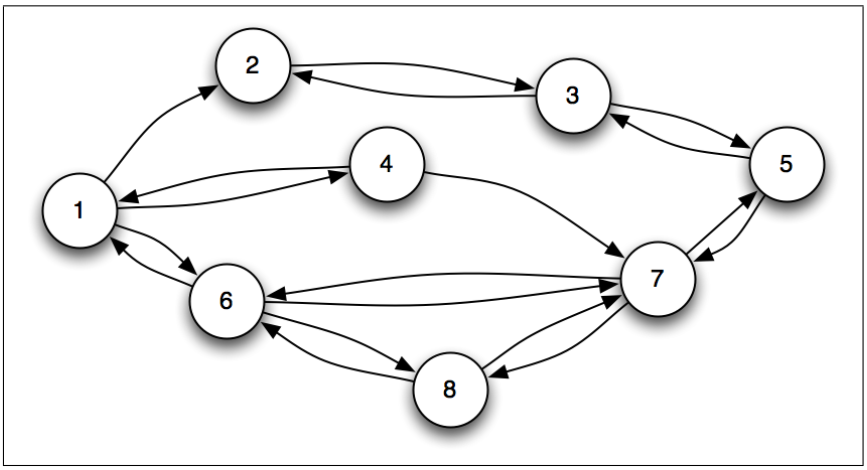

Figure 5.2: Example MAN.

5.2 Interaction Protocol

The previous section showed the different contents in the illocutions that agents can utter during the interaction protocol. In this section we focus on the protocol so that the reader may have a deeper comprehension of how the events described in Section 5.1 are ordered. Figure 5.1 gives an overview of the interaction protocol.

In this section we detail the interaction protocol through an example. Figure 5.2 represents a multiagent network that has been formed by agents certifying each other's identity. Each vertex in the graph represents an agent, and the number is its identifier (we will use a_i to refer to the agent with identifier i). An edge in the graph represents an identity certification that has not been cancelled. The edge is directed, and the starting point is the agent which uttered the identity certification, whereas the end point is the identity being certified. For example, the single edge between agents a_4 and a_7 means that there exists an identity certification illocution $\langle a_4, a_7, \langle 32, a_4, a_7\rangle\rangle$ with request identifier 32 in the global environment and no associated certificate cancellation. Notice that some agent pairs have two bidirectional edges. This will be the common case in which both agents have certified each other's identity.

In the example interaction protocol, agent a_1 interacts with agent a_5. The interaction protocol starts when the initiator agent (*i.e.,* a_1) sends an interaction request illocution with a_5 as its target. Agent a_1 cannot send a request directly to a_5 because the request would not be correct, since a_5 has not certified a_1's identity. An attempt to do so would get the request discarded. Therefore, a_1 may send a correct request to either a_4 or a_6 (not to a_2 since it hasn't certified a_1's identity either. If agent a_1 chooses a_4, the request illocution being sent would be: $\langle a_1, a_4, \langle 225, a_1, a_5, \langle\langle a_4\rangle, \emptyset, \emptyset\rangle\rangle\rangle$, assuming that a_1 has never used 225 as a request identifier before.

Upon reception of the illocution, agent a_4 verifies that it is correct for it. Nonetheless, since a_1 is the only agent that has certified its identity, it cannot create an entailing request that is correct by forwarding. Therefore, it creates an entailing request that is a re-route of the one it received, and sends the request back to a_1. This request would be: $\langle a_4, a_1, \langle 225, a_1, a_5, \langle\langle a_4\rangle, \emptyset, \{a_4\}\rangle\rangle\rangle$.

Agent a_1 receives the request illocution and verifies that it is correct. Now agent a_1 can only forward the request to agent a_6. It creates an entailing request illocution and sends it to agent a_6: $\langle a_1, a_6, \langle 225, a_1, a_5, \langle\langle a_4, a_6\rangle, \emptyset, \{a_4\}\rangle\rangle\rangle$. Agent a_6 receives the request illocution and verifies that it is correct, and it can forward to agents a_7 and a_8. When a_6 chooses to forward to a_8 it sends the following request: $\langle a_6, a_8, \langle 225, a_1, a_5, \langle\langle a_4, a_6, a_8\rangle, \emptyset, \{a_4\}\rangle\rangle\rangle$.

Let us assume that when a_8 receives the request, even though it is correct for it, it chooses not to forward the request towards the target but to block it (this is an essential part of enforcement). In this case, a_8 creates an entailing request which it sends back to a_6: $\langle a_8, a_6, \langle 225, a_1, a_5, \langle\langle a_4, a_6, a_8\rangle, \{a_8\}, \{a_4\}\rangle\rangle\rangle$.

Upon reception of the illocution, a_6 verifies that it is correct and creates an entailing request that is forwarded to a_7, the only agent it can still forward the request to. The request illocution is: $\langle a_6, a_7, \langle 225, a_1, a_5, \langle\langle a_4, a_6, a_8, a_7\rangle, \{a_8\}, \{a_4\}\rangle\rangle\rangle$. Agent a_7 verifies that the received request is correct for it and, since its identifier has been certified by the target agent a_5, chooses to forward an entailing request to the target: $\langle a_7, a_5, \langle 225, a_1, a_5, \langle\langle a_4, a_6, a_8, a_7, a_5\rangle, \{a_8\}, \{a_4\}\rangle\rangle\rangle$.

When a_5 receives the request illocution for which it is the target, it also verifies that the request is correct for it. After doing so, it chooses to interact with the initiator agent and informs all the agents in the request route about this by sending an acknowledgement illocution to each containing the request information in the request illocution that it received. Below we list all the acknowledgement illocutions that a_5 sends:

- $\langle a_5, a_7, \langle 225, a_1, a_5, \langle\langle a_4, a_6, a_8, a_7, a_5\rangle, \{a_8\}, \{a_4\}\rangle\rangle\rangle$.
- $\langle a_5, a_6, \langle 225, a_1, a_5, \langle\langle a_4, a_6, a_8, a_7, a_5\rangle, \{a_8\}, \{a_4\}\rangle\rangle\rangle$.
- $\langle a_5, a_4, \langle 225, a_1, a_5, \langle\langle a_4, a_6, a_8, a_7, a_5\rangle, \{a_8\}, \{a_4\}\rangle\rangle\rangle$.
- $\langle a_5, a_1, \langle 225, a_1, a_5, \langle\langle a_4, a_6, a_8, a_7, a_5\rangle, \{a_8\}, \{a_4\}\rangle\rangle\rangle$.

Notice that the target agent has not sent an acknowledgement illocution to the blocking agent a_8. Therefore, a_8 does not know whether a consensual interaction can take place.

After a_1 (*i.e.,* the initiator) checks that the acknowledgement illocution is justified (see Definintion 5.1.8), both the initiator and target start a joint action through a private communication channel: $\langle 225, a_1, a_5\rangle \in \mathcal{I}$. The model defines no structure for the contents of the joint action, since it only focuses on what goes one prior to the joint action and after it, when agents can send feedback about the joint action. This feedback is sent as the contents of an illocution. In our example we assume agent a_1 complains about the joint action by sending the following feedback illocutions:

- $\langle a_1, a_7, \langle\langle 225, a_1, a_5, \langle\langle a_4, a_6, a_8, a_7, a_5\rangle, \{a_8\}, \{a_4\}\rangle\rangle, a_1, 0\rangle\rangle$.
- $\langle a_1, a_8, \langle\langle 225, a_1, a_5, \langle\langle a_4, a_6, a_8, a_7, a_5\rangle, \{a_8\}, \{a_4\}\rangle\rangle, a_1, 0\rangle\rangle$.
- $\langle a_1, a_6, \langle\langle 225, a_1, a_5, \langle\langle a_4, a_6, a_8, a_7, a_5\rangle, \{a_8\}, \{a_4\}\rangle\rangle, a_1, 0\rangle\rangle$.
- $\langle a_1, a_4, \langle\langle 225, a_1, a_5, \langle\langle a_4, a_6, a_8, a_7, a_5\rangle, \{a_8\}, \{a_4\}\rangle\rangle, a_1, 0\rangle\rangle$.

We point the reader to the fact that agent a_8 received a feedback illocution even though it never received an acknowledgement illocution. This poses no problem, since the definition of a legitimate feedback does not depend on having received the acknowledgement. Upon receiving the feedback illocution, a_8 has access to the request that reached its target.

We also point out to the reader that no feedback has been sent to the target agent by the complaining initiator (the value 0 indicates the lowest satisfaction possible). It may be the case that the complaining agent does not want the other partner to know that it is complaining. Nonetheless, any of the recipients of the complaint could decide to forward the complaint to the other partner via a feedback illocution, *e.g.,* $\langle a_7, a_5, \langle\langle 225, a_1, a_5, \langle\langle a_4, a_6, a_8, a_7, a_5\rangle, \{a_8\}, \{a_4\}\rangle\rangle, a_1, 0\rangle\rangle$, thus letting the partner know about the feedback.

The last step of the interaction process comes after receiving the feedback illocution. The agent will verify whether it is legitimate (see Definition 5.1.10) so that no illegitimate complaints are taken into account when making decisions about future actions.

5.3 Behavioural Model

In Section 5.1 we have introduced our previous model adapted to the modified assumptions addressed in this chapter. These differences also bring about changes to the behavioural model, which we detail below. The obvious change is the reduction in the number of functions that model the behaviour of agents, since we have encapsulated all events into illocutions. Furthermore, another change is that the joint action among agents is not modelled. The reason being that we are not interested in what happens in the joint actions because the model does not define whether they are satisfactory. This is an effect of there not being any global norm, but a subjective satisfaction defined by each agent.

In the presented behavioural model we define a function describing the observable behaviour among agents.

Definition 5.3.1. Given a set of agent identifiers A, and a set of content identifiers I, a *behaviour* is a function $\iota : A \times A \times E \to \mathbb{I} \cup \{\bot\}$ that models the agents' illocution sending behaviour. Given the agent identifiers $a, a' \in A$ and an environment $e \in E$, $\iota(a, a', e)$ is the illocution that is sent from a to a' when the environment is e. If $\iota(a, a', e) = \bot$, no illocution is sent. Let B be the set of all agent behaviours.

The system behaviour describes how the environment (see Definition 5.1.2) evolves. The environment is a dynamic element that changes as agents interact with one another by sending illocutions. A behaviour defines which illocution is sent between a pair of agents given an environment. Therefore, since the system starts with an empty environment (an empty set of illocutions), the behaviour function describes the steps through which new illocutions are uttered and thus added to the environment. Mathematically speaking $e' = e \cup \bigcup_{a,a' \in A} \{\iota(a, a', e)\}$, where e is the original environment that is being updated into e'.

5.3.1 Behavioural Properties

As in Chapter 4 we can extract some analytical properties from the model. The current model can be seen as an extension of the one in Chapter 4, thus, some of the properties for the previous model are also valid for the current one, but some of them are not any more.

In Chapter 4, a global norm was given that described the actions that were valid under a specific environment. This norm was the same for all agents, and since all agents had access to the norm definition they all knew whether the actions they executed were normative. In this chapter, there is no such global definition of what is normative. Instead each agent has a subjective valuation of what is satisfactory for it. Therefore, what is satisfactory for one agent does not need to be satisfactory for another.

The properties in Chapter 4 defined different upper bounds on the number of norm violations depending on the behavioural properties of the agents in the MAN. Given that the current model does not have norm violations, we adapt some of the properties to define an upper bound on the number of feedback illocutions where the valuation denotes an unsatisfactory interaction. For the remainder of the section we will restrict the values of feedback to either 0, which denotes an unsatisfactory interaction, and 1, which denotes a satisfactory interaction. The model, however, allows for intermediate values.

Definition 5.3.2. A feedback $\langle\langle j, a_i, a_t, p\rangle, a_f, v\rangle$ is a *complaint* if $v = 0$. Let the predicate **complaint** $\subseteq F$ hold when a feedback is a complaint.

The current MAN is formed as an overlay network, *i.e.,* a virtual contact network on top of a communication network. Therefore, any peer can send illocutions to any other peer. Nonetheless, the recipient peer can choose to discard those illocutions with content that do not follow the restrictions defined in Section 5.1. By discarding those illocutions, the agent is exhibiting an enforcing behaviour.

Definition 5.3.3. Given an environment $e \in E$, an agent with identifier $a \in A$ is said to be *networked* if it discards from its local environment those interactions containing: requests that are not correct, acknowledgements that are not justified, and feedback that is not legitimate. We assume that all agents in a MAN, as defined in this chapter, are networked.

Now we give a definition for incompatible agents. An incompatible behaviour occurs when after a consensual interaction, either of the interacting partners sends a feedback message which denotes an unsatisfactory interaction (*i.e.,* a complaint). Two agents are incompatible when one of them has sent a complaint about the other.

Definition 5.3.4. Given an environment $e \in E$, two agents with identifiers $a, a' \in A$ are said to be incompatible to an agent with identifier $a'' \in A$, if a complaint that is legitimate to a'' has been sent about an interaction between a and a', *i.e.,* $\exists \langle a_s, a_r, f\rangle \in e$ $(f = \langle\langle j, a_i, a_t, p\rangle, a_f, v\rangle \wedge \textbf{legitimate}(f, a'', e) \wedge ((a_i = a \wedge a_t = a') \vee (a_i = a' \wedge a_t = a)) \wedge \textbf{complaint}(f))$. Let the predicate **incompatible** $\subseteq A \times A \times E$ hold when the two given agents are incompatible for the given local environment of the third agent.

We will now adapt the avoiding behaviour. In the current model an avoiding behaviour occurs when an agent that has legitimately complained about another will never acknowledge requests coming from it. Again, this is an enforcement mechanism that can be applied by any agent.

Definition 5.3.5. Given an environment $e \in E$, a behaviour function ι is said to be *incompatible avoiding* for an agent with identifier $a \in A$ when a never sends requests with it as initiator and an incompatible agent as target, and never acknowledges requests coming from agents that are incompatible with it, *i.e.,* $\textbf{incompatible}(a, a', e) \rightarrow (\iota(a, a'', e) \neq \langle a, a'', \langle j', a', a, p'\rangle\rangle) \wedge \iota(a, a'', e) \neq \langle a, a'', \langle j', a, a', p'\rangle\rangle)$. Let the predicate **avoiding** $\subseteq B \times A$ hold when a behaviour function is incompatible avoiding for a given agent.

For the following proofs we define the function $\psi : A \times A \times E$. Given two agent identifiers $a, a' \in A$, and an environment $e \in E$, $\psi(a, a', e)$ is the set of legitimate complaints by a about a', *i.e.,* $\psi(a, a', e) = \{\langle\langle j, a_i, a_t, p\rangle, a, v\rangle \in F \mid \textbf{complaint}(\langle\langle j, a_i, a_t, p\rangle, a, v\rangle) \wedge \textbf{legitimate}(\langle\langle j, a_i, a_t, p\rangle, a, v\rangle, a, e) \wedge ((a_i = a \wedge a_t = a') \vee (a_t = a \wedge a_i = a'))\}$.

Lemma 5.3.6. *If the behaviour function ι is incompatible avoiding for agent a, the environment will contain at most one complaint originated at a about another agent a', i.e.,* $\textbf{\textit{avoiding}}(\iota, a) \rightarrow |\psi(a, a', e)| \leq 1$.

Proof. Let us assume that $\textbf{avoiding}(\iota, a) \wedge |\psi(a, a', e')| = 1$, that is, there exists a legitimate complaint in environment e' issued by a about a', where e' represents the environment at a moment prior to e, thus, $e' \subseteq e$. According to definition 5.3.5, a will never send a request as initiator with a' as target, and it will never acknowledge a request with a' as initiator. Therefore, no complaint sent by a about a' is legitimate (see Definition 5.1.10). Which proves that $\psi(a, a', e)$ can never contain more than one legitimate complaint by a about any other agent. □

Theorem 5.3.7. *If the behaviour ι of a multiagent network is incompatible avoiding for all agents, then there exists an upper bound to the total number of legitimate complaints in the environment $e \in E$. This upper bound is twice the number of certified agent identity pairs, i.e., $|A_e|(|A_e| - 1)$.*

Proof. The total number of complaints in a multiagent network is equal to the sum of complaints issued by each agent. Furthermore, an agent can only issue legitimate complaints when the initiator's identity has been certified by someone, otherwise its requests would be discarded as they would not be correct (see Definition 5.1.7). Therefore, one can calculate the total upper bound of legitimate complaints by adding the upper bounds for each agent with a certified identifier.

$$\sum_{a,a' \in A_e} \psi(a, a', e)$$

Given that the behaviour is incompatible avoiding for all agents in the system, Lemma 5.3.6 applies to all agents. Therefore,

$$\sum_{a,a' \in A_e} \psi(a, a', e) \leq |A_e|(|A_e| - 1)$$

which proves the theorem. □

The following result may be proved in much the same way as Lemma 5.3.6 and Theorem 5.3.7:

Corollary 5.3.8. *Given a behaviour $\iota \in B$ that is violator avoiding for a group of agents $A' \subseteq A$ (i.e., $\forall a \in A'$ (**avoiding**(ι, a))), then there exists an upper bound to the total number of legitimate complaints by an agent in A' that can exist in the environment. This upper bound is $|A_e|(|A'| - 1)$.*

These results are very similar to those in Chapter 4. Nonetheless, there is an issue in applicability. Since the multiagent network in this chapter is open (*i.e.,* new identities can join and leave), an upper bound based on the number of certified identifiers is not meaningful. An agent can create any number of identifiers and certify them. This agent would only need one other agent in the network to certify one of its identifiers in order to use all of the ones it has created. This type of behaviour is called a Sybil attack in the literature [Douceur, 2002]. Since such an attack is possible under the model in this chapter, the upper bound on the number of legitimate complaints is not meaningful. In Section 5.4.7, we show that it is possible to modify a reputation mechanism to work on top of this network so that it is Sybilproof. Furthermore, we show that by using the SybilLimit techniques [Yu et al., 2008] we find a meaningful upper bound on the number of legitimate complaints.

5.4 Avoiding Fraud

Peers in the network can attempt fraud in many different ways. For one, an agent can potentially send any illocution content it wants. In Section 5.1 we have seen some checks that agents can do in order to avoid some types of illocution content fraud, such as checking for correctness in interaction requests, checking for justification in acknowledgements, and checking for legitimacy in

feedback. Nonetheless, an agent could easily fake a correct interaction request so that the subsequent acknowledgement and feedback are justified and legitimate, respectively. Furthermore, since a reputation mechanism is coupled with the enforcement mechanism, an agent or group of agents can also attempt to correctly follow the protocol with the purpose of subverting the reputation mechanism in order to get chosen to interact more often than they ought to. This can be done by either raising one's reputation or by lowering that of peers that have a higher reputation rank. The following is a non-exhaustive list of attacks: badmouthing, *i.e.,* sending misleading feedback about prior interactions; ballot-stuffing, *i.e.,* sending feedback of non-existing interactions; milking or dynamic personality, *i.e.,* getting good assessments by satisfying other peers in order to cheat later on; collusion, *i.e.,* forming a collective with other peers to try to subvert the assessment system; whitewashing, *i.e.,* changing the peer's identifier in order to avoid negative assessments from previous feedback; and finally, Sybil, *i.e.,* creating many false identities controlled by an agent that collude to subvert the reputation system.

In the following sections we go into more detail for each of the fraud strategies described above. For each of these fraud strategies we detail the way in which they would be attempted by malicious agents in our system. Furthermore, we describe techniques that can be used to lessen the effectiveness of the attack in subverting the system

5.4.1 Data Fraud

This section tackles the case in which an agent sends false illocution contents (*i.e.,* illocution contents that do not follow the interaction protocol guidelines) in order to corrupt the local environments of other agents with information that benefits it.

To show how this could be done, let us take the example in Section 5.2. Imagine agent a_8 (the one that blocked the request in the example) wants to spread bad feedback about agent a_1, *i.e.,* badmouthing. Agent a_8 could send a fake interaction request illocution with a_1 as the initiator and a non-existent agent a_{32} as the target $\langle a_8, a_7, \langle 123, a_1, a_{32}, \langle\langle a_8, a_7\rangle, \emptyset, \emptyset\rangle\rangle\rangle$. Since the target agent does not exist, no route would ever be found and then a re-routing request would eventually reach a_8 (*e.g.,* $\langle a_7, a_8, \langle 123, a_1, a_{32}, \langle\langle a_8, a_7, ..., a_4\rangle, \emptyset, \{a_7, ..., a_4\}\rangle\rangle\rangle\rangle$). Now the agent a_8 can send legitimate feedback illocutions to all the agents in the request route that got re-routed (*e.g.,* $\langle\langle 123, a_1, a_{32}, \langle\langle a_8, a_7, ..., a_4, a_{23}, a_{32}\rangle, \emptyset, \{a_7, ..., a_4\}\rangle\rangle, a_{32}, 0\rangle$). That is, a_8 makes it appear as if the request eventually reaches a_{32} via another non-existing agent a_{23}.

Another possible fraud is to tamper with interaction requests. Possible tampering is: adding or removing agents from a recipient list, adding or removing agents from the blocking set, adding or removing agents from the re-route set, and changing the initiator or target agents. The first three tampering examples could be done to add or remove blame for helping or hindering request routing. The last example could be useful for man in the middle attacks, which

is a form of active eavesdropping in which the malicious agent makes independent connections with the victims and relays messages between them, making them believe that they are interacting directly with each other over a private connection, when in fact the entire conversation is controlled by the attacker. Finally, a typical fraud would be that of identity fraud, by which an agent sends illocutions making believe it is another agent.

We tackle these types of fraud through typical encryption and signature techniques. Each agent creates its own encryption key pair $\langle a_i, p_i \rangle$. The first key is public, being this the agent's public identifier. The second key is private. Note that we do not use a Public Key Infrastructure (PKI), as it is a centralised component which could suffer from denial of service attacks, and it is a bottleneck. Instead we use Pretty Good Privacy (PGP), in which each agent creates its own identity and it is other agents that certify it. The chance of two agents generating the same key pair is infinitesimal with large enough keys.

In order to tackle identity fraud, the illocutions are encrypted with the private key of the sender and re-encrypted with the public key of the receiver. Upon reception of an illocution, the receiver decrypts the message with its private key and then decrypts it with the sender's public key. This ensures that the illocution sender has access to the private key associated to the sender's identity, and that only an agent that has access to the recipient's private key can read the illocution contents. If agents are careful not to allow others to know their private keys, the previous encryption technique ensures authentication, integrity, and privacy.

Tackling interaction request fraud is in change solved via digital signatures, as opposed to encryption. Encryption is not useful since we want any agent to be able to read the contents of the interaction request. By using digital signatures, we can authenticate the illocution's origin and test its integrity. Since the request route of a request is constructed during the route process, the signature scheme is more complex. Each interaction request is sent with n associated digital signatures, where n is the number of entailing interaction requests for the current request. The content being signed by each signature is the entailing interaction request that was sent by the agent signing it. To show how it would work, we will use the example in Section 5.2. For this example, let S be the set of all digital signatures, and $\sigma : R \times A \to S$ be the signature function. Given a request $r \in R$ and an agent identifier $a \in A$ (which is also the agent's public key), $\sigma(r, a)$ is a's digital signature of r.

The following list shows each requests sent during the routing process, and the signatures that where associated to each request:

- $r_1 = \langle 225, a_1, a_5, \langle \langle a_4 \rangle, \emptyset, \emptyset \rangle \rangle$; $\sigma(r_1, a_1)$
- $r_2 = \langle 225, a_1, a_5, \langle \langle a_4 \rangle, \emptyset, \{a_4\} \rangle \rangle$; $\sigma(r_1, a_1)$, and $\sigma(r_2, a_4)$
- $r_3 = \langle 225, a_1, a_5, \langle \langle a_4, a_6 \rangle, \emptyset, \{a_4\} \rangle \rangle$; $\sigma(r_1, a_1)$, $\sigma(r_2, a_4)$, and $\sigma(r_3, a_1)$
- $r_4 = \langle 225, a_1, a_5, \langle \langle a_4, a_6, a_8 \rangle, \emptyset, \{a_4\} \rangle \rangle$; $\sigma(r_1, a_1)$, $\sigma(r_2, a_4)$, $\sigma(r_3, a_1)$, and $\sigma(r_4, a_6)$

- $r_5 = \langle 225, a_1, a_5, \langle\langle a_4, a_6, a_8\rangle, \{a_8\}, \{a_4\}\rangle\rangle$; $\sigma(r_1, a_1)$, $\sigma(r_2, a_4)$, $\sigma(r_3, a_1)$, $\sigma(r_4, a_6)$, and $\sigma(r_5, a_8)$

- $r_6 = \langle 225, a_1, a_5, \langle\langle a_4, a_6, a_8, a_7\rangle, \{a_8\}, \{a_4\}\rangle\rangle$; $\sigma(r_1, a_1)$, $\sigma(r_2, a_4)$, $\sigma(r_3, a_1)$, $\sigma(r_4, a_6)$, $\sigma(r_5, a_8)$, and $\sigma(r_6, a_6)$

- $r_7 = \langle 225, a_1, a_5, \langle\langle a_4, a_6, a_8, a_7, a_5\rangle, \{a_8\}, \{a_4\}\rangle\rangle$; $\sigma(r_1, a_1)$, $\sigma(r_2, a_4)$, $\sigma(r_3, a_1)$, $\sigma(r_4, a_6)$, $\sigma(r_5, a_8)$, $\sigma(r_6, a_6)$, and $\sigma(r_7, a_7)$

Given an interaction request, it is easy to calculate all the entailing requests and what agent identifier should have signed each. Therefore, given a request and a set of signatures, any agent can verify whether there has been any tampering. Sending one digital signature for each entailing request is an overhead that should be taken into account. The FIPS 186-3 standard for encryption using the DSA digital signature algorithm, recommends using public keys that are 2816 bits which generate signatures 4096 bits long. When using the recommended values for the previous example, the request r_7 would be 25408 bits long (assuming 64 bit request identifiers) and its associated signatures would be 28672 bits long. Therefore, the signature scheme more than doubles the length of the messages. Nonetheless, this overhead brings about two benefits. Firstly, it allows authentication and integrity. More importantly, it makes storage of sent and received interaction request illocutions unnecessary, since all the entailing illocutions can be calculated given an illocution, and the associated signature provides authentication. The entailing interaction request illocutions are needed in order to test for correctness of interaction requests, justification of acknowledgements, and legitimacy of feedback.

The same type of fraud avoiding technique is applied to acknowledgements and feedback, since both contain an interaction request. In case of an acknowledgement a new signature is added by the target agent, which authenticates that the target has received the interaction request. Continuing with the example, the acknowledgement $k_1 = \langle a_5, a_1, \langle 225, a_1, a_5, \langle\langle a_4, a_6, a_8, a_7, a_5\rangle, \{a_8\}, \{a_4\}\rangle\rangle\rangle$ would have the associated signatures $\sigma(r_1, a_1)$, $\sigma(r_2, a_4)$, $\sigma(r_3, a_1)$, $\sigma(r_4, a_6)$, $\sigma(r_5, a_8)$, $\sigma(r_6, a_6)$, $\sigma(r_7, a_7)$, and $\sigma(r_7, a_5)$. The last one being the signature by the target agent which certifies that the interaction request has reached it and it accepts to interact via the acknowledgement.

Feedback also needs to be authenticated and tested for integrity. This is done by adding a signature by the agent giving the feedback to the set of signatures in the acknowledgement. Continuing with the example above, the feedback $f_1 = \langle\langle 225, a_1, a_5, \langle\langle a_4, a_6, a_8, a_7, a_5\rangle, \{a_8\}, \{a_4\}\rangle\rangle, a_1, 0\rangle$ sent by agent a_1 to agent a_8 in the illocution $\langle a_1, a_8, f_1\rangle$ will have the associated signatures $\sigma(r_1, a_1)$, $\sigma(r_2, a_4)$, $\sigma(r_3, a_1)$, $\sigma(r_4, a_6)$, $\sigma(r_5, a_8)$, $\sigma(r_6, a_6)$, $\sigma(r_7, a_7)$, $\sigma(r_7, a_5)$, and $\sigma(f_1, a_1)$. The last one is the signature by the agent giving the feedback which certifies the authenticity and integrity of the feedback.

At each step of the interaction process the illocution being sent contains all the signatures of the previous steps. This is a large overhead that allows authentication and integrity testing. Furthermore, sending all the signatures reduces

the need for memory storage by the agents, since the previous steps of an interaction process can be calculated from the current one (*e.g.,* the acknowledgement request can be extracted from the feedback, and all the entailing interaction requests can also be calculated). Therefore, an agent can check whether an acknowledgement is justified without having to store the previous interaction request illocutions it sent. The same holds for agents checking whether a feedback illocution is legitimate.

5.4.2 Badmouthing

A badmouthing attack occurs when an agent sends negative feedback about interactions for which it has no cause to complain. Through this attack, an agent tries to lower its competitors' reputation in order to gain relative reputation itself. This attack is a big issue when the trust assessment mechanism is said to be symmetric, *i.e.,* when the trust on an agent is the same independently of the agent assessing it. If the trust metric is asymmetric, *i.e.,* trust measures are subjective, badmouthing can be tackled. Furthermore, if the assessment mechanism only uses feedback about those interactions for which it was a partner, badmouthing will only affect his own future assessments. Nonetheless, limiting the feedback to be used in calculating trust to that of the assessing peer can cause bad assessments due to the lack of data. Some assessment algorithms use feedback from third parties to make up for the lack of data, which opens the door to badmouthing attacks. Some of these algorithms incorporate mechanisms that make them robust to badmouthing. What these mechanisms do is to give more weight to feedback coming from agents that give feedback similar to oneself or to those agents that have a good reputation themselves. Notwithstanding, the latter case assumes that peers with good reputation will always give good feedback which need not necessarily hold.

5.4.3 Ballot-Stuffing

Ballot-stuffing is an attack by which an agent sends more feedback than interactions it has been partner in. The interaction protocol in Section 5.2 together with the encryption techniques discussed in Section 5.4.1 are enough to ensure that a single agent cannot realise a ballot-stuffing attack. Nonetheless, a group of agents can realise a ballot-stuffing attack by executing the interaction protocol many times. In order to spread feedback as much as possible, they have to get as many agents in the interaction request route as possible, and this is out of their control. Since the routing algorithm is a variation of the shortest path search, (see Section 5.8) the agents attempting ballot-stuffing attacks would have to be far apart in the network, which is unlikely. Notwithstanding, assuming that the malicious collective can manage to be far apart, and spread their ballot-stuffed feedback, there are techniques that can make a trust assessment algorithm robust to ballot-stuffing. The main counterattacks used in the literature are: filtering feedback that comes from peers suspect to be ballot-stuffing

as in Section 5.4.2, and using feedback per interaction rates instead of accumulation of feedback [Kamvar et al., 2003, Tian et al., 2008, Xiong et al., 2004].

5.4.4 Dynamic Personality

An agent that achieves a high reputation will be able to interact more often. In which case it can attempt to deceive other agents taking advantage of its high reputation. This is called milking the reputation, or having a dynamic personality. If the assessment algorithm takes long to adapt to changing strategies, it can suffer from dynamic personality attacks. This is specially so with assessment mechanisms that take into account all the past history of an agent in order to calculate its reputation.

A commonly used technique to make the reputation mechanism robust to dynamic personality attacks is to have a memory window so that not all the past history is taken into account. An even more robust mechanism is to have a dynamic memory window which is shortened when the reputation is lowered [Xiong et al., 2004]. This reduces the amount of reputation milking that can be achieved.

5.4.5 Whitewashing

Whitewashing occurs when an agent changes its identifier in order to escape previous bad feedback. Simple whitewashing, in which an agent changes its identity without the collaboration of others is not an issue for the system we have described independently of the reputation mechanism being used as long as peers are networked (see Definition 5.3.3). This is so because an agent which changes its identifier also looses the certifications from its former contacts, which no longer recognise it. Therefore, the cost of getting new contacts or of convincing the old contact to certify its new identity ought to be enough to discourage whitewashing attacks. Nonetheless, a group of malicious agents could collude to allow whitewashing by certifying new identities. This is no different than a Sybil attack (see Section 5.4.7).

5.4.6 Collusion

Collusion occurs when a group of agents co-operate with one another in order to take advantage of the system and other agents. In our case it means that agents in the colluding group will always give each other good ratings. As with badmouthing, filtering out feedback from colluding peers seems to be the most used way to make a reputation mechanisms robust against collusion attacks. The techniques used for robustness against badmouthing can also be used in this case: filtering feedback coming from peers whose feedback is not similar to one's own. Furthermore, the MAN model in this chapter allows collusion to be detected by the routing mechanism. The paths between colluding peers will always share most of the routing agents. Therefore, the feedback from collusion attacks will be concentrated in the part of the network through which

the colluders are linked[1], thus it is not easy for the colluders to corrupt large parts of the network.

5.4.7 Sybil Attacks

Sybil attacks are a sort of security threat that can be launched when identifiers are cheap. The attack consists in creating enough identities so that a single agent can subvert the normal functioning of the system. Some examples of a Sybil attack are taking control of specific portions of a distributed hash table (DHT), or improving one's reputation through false feedback from many Sybil identities.

The main ways in which such attacks are countered are either by adding a cost to creating identifiers or by having a central authority that grants identities. The problem with the first approach is that non Sybil identities also incur this cost which may degrade the usage of the platform. The latter approach goes against the distributed nature of P2P and MAS.

There are other approaches that reduce the amount of Sybil identities that can be used. SybilGuard [Yu et al., 2006] and SybilLimit [Yu et al., 2008] both use a contact network with specific algorithms for checking identities that guarantee that no more than $\Omega(\sqrt{n}\log n)$, and $\Omega(\log n)$ Sybil identities are accepted by any node, respectively (where n is the total number of honest identities). Furthermore, [Cheng and Friedman, 2005] give a description of the types of distributed reputation algorithms that are SybilProof, that is, that no agent can increase its reputation (in absolute or relative terms) by launching a Sybil attack. Sybilproof mechanisms are built through transitive trust in a contact network.

The network we propose in this thesis is a contact network, and as such it shares the same base commonality with the three previous mechanisms, namely SysbilGuard, SybilLimit, and SybilProof. Specifically SybilGuard and its improvement SybilLimit can be implemented directly into our architecture to reduce the number of Sybil nodes that are accepted. Nonetheless, since the enforcement mechanisms are coupled with reputation assessment mechanisms, we are also interested in countering reputation attacks. In the current section we show how reputation mechanisms built on top of the proposed network can be made Sybilproof. A reputation mechanism is Sybilproof if the function it uses to calculate the reputation of an agent, taking as input the values of the edges in the contact graph, satisfies the conditions in the following theorem extracted from [Cheng and Friedman, 2005]:

Theorem 5.4.1. *A reputation calculating function is of the form* $f(s,t) = max_{P \in P_{s,t}} g(P)$*, where* g *is a function that goes from paths to real numbers. The function calculates the reputation of an identity* t *from the point of view of* s*. Such function is Sybilproof if:*

[1] It is likely that colluders will have contact links to one another. Nonetheless, they could still route through other links in order to try to influence a larger part of the network when realising this attack.

1. *(Diminishing returns) For all paths p, if p' is an extension of p, then $g(p') \leq g(p)$.*

2. *(Monotonicity) g is nondecreasing with respect to the edge values.*

3. *(No splitting) Given a single path p from s to t, if we split p into two paths p_1 and p_2 each from s to t, then $max(g(p_1), g(p_2)) \leq g(p)$ (the theorem assumes that an edge in the graph with an associated value v can be split into two edges with associated values v_1 and v_2 respectively, as long as $v = v_1 + v_2$).*

5.5 Existing Reputation Mechanisms

This section describes the most influential reputation assessment algorithms in the literature. For each of these algorithms we analyse their robustness against the malicious attacks in Section 5.4, and the complexity measured in messages that they send.

5.5.1 Bin Yu and Munindar Singh

Yu and Singh's work [Yu and Singh, 2000] is one of the first in the area of distributed reputation systems. They use a network of trust relations in order to propagate the reputation measures. Agents rate other agents after direct interactions (becoming witnesses of one another), forming an interaction trust graph. In order to assess the reputation of an agent with whom one has not interacted directly, the paths in the trust graph are used to calculate the reputation. Yu and Singh propose a simple algorithm to aggregate these trust measures into a reputation measure. This mechanism takes into account the reputation of the agents rating the agent being assessed. This follows from the assumption that peers that act positively will give positive feedback. This assumption opens the door to reputation attacks, such as collusion, badmouthing, and Sybil attacks.

Their approach does not tackle many of the known attack schemes since it was not in their scope to tackle these issues. They even admit that their approach does not incentive co-operation among agents, since a bad initial reputation is confirmed through negative actions. Badmouthing, ballot-stuffing, and collusion attacks are not dealt with. Nonetheless, they do tackle the dynamic personality issue by giving a larger weight to negative interactions than to positive interactions. Thus making an agent gain reputation slowly but loose it quickly.

Furthermore, they do not mention how agents are identified, but they do describe their system as a totally distributed one. We assume that no certification authority is present, thus making whitewashing and Sybil attacks possible. Their approach is definitely not robust against these attacks. Firstly because negative reputations are possible and a new agent has an initial reputation of 0, which makes whitewashing attacks fruitful for malicious entities. Secondly, if identifiers are cheap, a Sybil attack is possible. The system presented cannot

be shown to be Sybilproof since it does not satisfy the no splitting property in Theorem 5.4.1 of Sybilproof systems.

Finally, their approach seems to be very costly, since the reputation values are spread through the network through a flooding policy. Furthermore, the turnover issue is not mentioned in their work. Nonetheless, an analysis of their work shows that the reputation values would not degrade much from agent turnover since the maximum valued paths to the witnesses are used.

5.5.2 Aberer and Despotovic

Aberer and Despotovic propose a decentralized trust management system in [Aberer and Despotovic, 2001] where all feedback from interactions is stored in a P-Grid DHT [Aberer et al., 2003]. In order to avoid feedback loss from the probability of having a malicious peer manage the keys for the feedback about it, feedback is replicated accross multiple peers. The basis for their approach to detect malicious peers is that they will be complained about by many peers.

Their approach is robust to badmouthing attacks since filing complaints is detrimental to the peer's own reputation if other peers have been complaining about the peer. Although in their experiments, malicious peers do not execute badmouthing attacks, they just return random data when queried as witnesses.

Their approach can have problems with ballot-stuffing attacks, since every agent p can file a complaint about q at any time. By not providing a mechanism that checks that a complaint is legitimate, ballot-stuffing attacks can be executed easily.

Turnover is tackled by normalising the number of complaints depending on the frequency that the witnesses are found on the network. This normalisation makes the reputation calculating algorithm robust against peer turnover. Nonetheless, such normalisation does nothing to prevent ballot-stuffing attacks.

In [Aberer and Despotovic, 2001] no mention is done to whitewashing. From the analysis of their approach one can see that a peer would easily escape a bad reputation by changing its own identity. Furthermore, no mention is made about a collective of peers trying to subvert the system through collusion or a Sybil attack. A close look at their decision algorithm shows that the trust of the witnesses is assessed and the trust on a peer depends on the trust on the witnesses of interactions with that peer. This heuristic is based on the assumption that a reputed peer does not act maliciously when giving feedback. Since this assumption does not hold for collusion or Sybil attacks, we believe the approach described is not robust against these attacks.

The work in [Aberer and Despotovic, 2001] presents an attack by malicious peers that do not always cheat, just a percentage of the time given a cheating probability. The results from those experiments show that such malicious activity hinders the correct functioning of their algorithm. Such findings lead us to believe that their algorithm is not robust against dynamic personality attacks.

The main issue the approach by Aberer and Despotovic tackles is cost. Their algorithm uses a DHT based on P-Grid in order to store and retrieve the feedback from the peer whose trust is being assessed. In the complex algorithm the

DHT is queried $w+1$ times, where w is the number of witnesses (*i.e.,* partners) of interactions with the peer being assessed. Therefore, an interaction has a cost of $O((w+1)\log n)$ where n is the total number of peers. This approach has a cost that is larger than that of our approach by a small factor, since the number of witnesses is small compared to the number of agents in a system.

5.5.3 Eigentrust

Eigentrust [Kamvar et al., 2003] is a distributed algorithm to calculate the global transitive trust on each agent of the network. The transitive trust is calculated through the left principal eigenvector, which can be approximated via a random walk to reduce the complexity of the distributed algorithm. The algorithm is further modified to add a probability to crawl through the pre-trusted peers. Kamvar et al. assume that there will always exist a set of peers that are trustable and known to all. In order to further secure the algorithm, each reputation measure is stored in a replicated DHT, where each reputation value is stored by m peers. This way malicious peers cannot easily modify the results of the algorithm.

Eigentrust is robust to ballot-stuffing since they normalise the local trust values of agents. This normalisation spreads an agent's trust through all the other agents in the network so that all trusts add up to 1. Ballot-stuffing is impossible in this scenario.

Eigentrust does not tackle the whitewashing attack in a satisfactory way. All new peers have an initial trust value of 0. The article mentions two mechanisms to choose a partner peer: deterministic and probabilistic. In the deterministic approach, the peer with the highest reputation is chosen. This approach is robust against whitewashing attacks, but new peers are never allowed to gain reputation and the load balancing of the network is severely compromised. The probabilistic mechanism chooses a peer with a probability that is proportional to its reputation, with a 10% probability a peer with trust equal to 0 is chosen. This approach allows new agents to gain reputation and maintains the load balancing property while opening a small (*i.e.,* 10%) door to whitewashing attacks.

Turnover is not tackled by the Eigentrust article. Their experiments are run with a low number of peers that do not leave the network. The authors of the algorithm do mention that the DHT in which the reputation values are to be stored ought to have a mechanism so that these values are not lost when a peer leaves the system.

Collusion by a group of agents with badmouthing techniques is tackled through the use of a pre-trusted set of peers. In order to do so, the authors of Eigentrust assume that

> "... there are some peers in the network that are known to be trustworthy. For example, the first few peers to join a network [...], since the designers and early users of a P2P network are likely to have less motivation to destoy the network they built."

Later on they acknowledge that it is important that no pre-trusted peer be a member of a malicious collective, as this would compromise the algorithm. The effectiveness of the trust algorithm should not rest on the existence of pre-trusted peers.

The Eigentrust algorithm does not tackle dynamic personality attacks. Kamvar et al. do mention the posibilty of a peer returning a percentage of valid files in order to fool the reputation system, and their approach is robust against this type of attack. Nonetheless, since they use the whole history of feedback to calculate reputation, their algorithm is very likely to suffer from dynamic personality attacks.

The issue of Sybil attacks is bypassed by mentioning that a cost could be added to generating a new identifier, such as CAPTCHAs[von Ahn et al., 2003] of identifying the text in an image. Nonetheless, attackers have been known to bypass such measures by reposting them in fake sites so that humans trying to gain access can solve them. Furthermore, their algorithm is symmetric, which has been shown to not be Sybilproof.

As for the complexity of their approach, each time a peer searches for a file it must assess the reputation of all the peers that respond to its query. Since the reputation value is stored by m peers each reputation assessment consists of $2m \log n$ messages, where n is the total number of peers in the network.[2] Furthermore, each time feedback is given, another $m \log n$ messages have to be sent. Finally at the end of each round the reputation must be recalculated, at which point $rn \log n$ messages are sent to update the reputation values, where r is the number of rounds needed for the algorithm to converge (each round all peers send each other the current reputation values). Therefore, we can conclude that the complexity of Eigentrust is $O(n \log n)$ which is higher than our approaches (see Sections 5.6.1 and 5.6.2).

5.5.4 Reciprocative decision

In [Feldman et al., 2004a] the authors describe various mechanisms for assessing the reputation of other agents based on what they call "reciprocative decision functions". The reputation mechanism is aimed to work in file-sharing P2P networks in which an agent cannot know whether the server from which they are requesting a file has defected when it ignores the request. An interaction in their context consists of an agent requesting to download a file, and the server deciding whether to share it or not. The interaction outcomes have been modelled according to a generalised prisoner's dilemma (GPD) that is asymmetric but maintains the social dilemma.

A reciprocative decision function calculates the normalised generosity of the agent being assessed and compares it to its own generosity. The generosity is calculated as the ratio of the number of provided services to the number of consumed services by an agent. In order to tackle the different problems

[2] The number of messages needed to store and retrieve a value from a DHT is assumed to be $\log n$, as in CAN, Chord, or P-Grid.

associated to reputation mechanism, they propose some modifications to the basic reciprocative decision function.

The first problem is the difficulty of getting enough feedback about other agents due to the large number of agents and the inherent turnover. To solve this Feldman et al. propose a server selection mechanism in which agents store the list of agents they have served in order to select them in the future and allow for reciprocation to happen. The other solution they propose is having a shared history implemented through a DHT. In our approach the feedback is spread through the paths used to deliver interaction requests, after which we use a private history approach in which each agent has access to its own data. By spreading feedback in this way we also tackle the same problem without the inconvenient of deploying a shared history. These are the overhead in memory access and the possibility of shared history attacks.

The second problem they tackle is collusion via ballot-stuffing. As they mention in their article, a shared history is vulnerable to these types of attacks. They propose to use a subjective measure of reputation based on maxflow (*i.e.,* finding the path with the maximum reputation "flow"), thus making their algorithm robust to simple ballot-stuffing attacks. Nonetheless, the mechanism is not robust against more complex attacks such as the "mole" attack (*i.e.,* one agent provides correct service while ballot-stuffing about its colluding partners). By studying the maxflow algorithm, one can see that it is not Sybilproof as it adds the values of different paths instead of returning the maximum value. Our approach, which is a modification to maxflow that is Sybilproof, does not suffer as much from shared history attacks because it uses a private history approach and a fraud detection mechanism. Furthermore, we use the personal similarity measure to verify the validity of feedback, which prevents mole attacks. For a mole to be effective in our system, it would have to give feedback about other agents that is similar to ours and positive feedback about its colluding partners, which is very costly.

Another issue they tackle is badmouthing. In their system a badmouthing attack is detected by the difference in feedback from the interacting agents. This is easy in the system they study because there are only two outcomes: file provided, or file not provided. When such inconsistencies are found the feedback given by the agent closer to the assessing agent is believed. In our case, we tackle badmouthing through the similarity measure.

Feldman et al. also provide a mechanism for countering whitewashing attacks — the "Stranger Adaptive" policy. This policy decides whether to interact with a stranger (*i.e.,* an agent for which no feedback is found) based on previous interactions with strangers. The issue with this approach is that new agents joining contemporary to many whitewashing attacks will not receive cooperation. Our approach is more targeted, and only new agents in the same area as other whitewashers would suffer the consequences of such attacks.

Finally, they tackle dynamic personality attacks by maintaining a short-term history. This mechanism is also implemented in our approach with the extension of dynamic history windows as seen in [Xiong et al., 2004].

The work on the reciprocative decision function does tackle the complexity

issue lightly. They mention that the maxflow algorithm is $O(V^3)$, where V is the set of vertices in the graph, but they provide a heuristic algorithm that has a constant time. They do not mention the associated cost to operating a DHT. Assuming that the heuristic algorithm provides a quick answer, it would have to access the DHT as many times as nodes in the network it would check. No estimate is given for the number of times the DHT would be accessed in their work. Since the maxflow algorithm searches through all the paths, we could assume that the data associated to each node in the graph would be accessed. Assuming that the DHT has some sort of mechanism for duplicating data in order to counter the attacks of colluding agents, the cost in DHT messages per assessment would be $O(r \cdot n \log n)$, where r is the replication factor. In their system the only agent assessing the reputation is the server. Therefore the cost per interaction would be $O(r \cdot n \log n)$, which is larger than the one of our approach, which is $O(\ln n)$.

5.5.5 Peertrust

Peertrust [Xiong et al., 2004] is a comprehensive approach to a decentralised reputation algorithm. Xiong and Liu propose different techniques to calculate the reputation and compare them through experiments. Their personal similarity measure (PSM) is based on the assumption that the credibility of a peer's feedback is proportional to the similarity to one's own feedback. Instead of assuming that the credibility depends on the reputation as a partner. Therefore, their algorithms that use PSM are robust against badmouthing attacks and collusion among different peers.

Their algorithms are also robust against ballot-stuffing because they calculate the average satisfaction and not the accumulated satisfaction. Therefore, more quantity of feedback will not necessarily improve the reputation, it is the quality of feedback that counts. The algorithm uses a dynamic window in order to tackle dynamic personality attacks. The reputation is calculated for a short and a longer window of time. If the short window value goes below the long window vale, the short value is used. Therefore, reputation becomes hard to build up but easy to destroy.

Nonetheless, their approach cannot fight against whitewashing or Sybil attacks satisfactorily. Xiong and Liu assume that a certification authority (CA) exists, which introduces a centralised component that breaks the P2P scheme. Even if they used an identification scheme without the centralised component, they do not mention what the reputation algorithm returns for a peer without any feedback. If the value returned is the minimum possible reputation, the whitewashing issue would be tackled but new peers would never be able to gain any reputation since nobody would ever interact with them. In Section 5.6.1 we present a modification of this algorithm that is made robust to whitewashing attacks. As for Sybil attacks, Peertrust PSM algorithms may have a good chance since they do pretty well with up to 70% of colluding peers. Nonetheless, a successfull Sybil attack could reach a higher percentage of the network, for which no information is given. Furthermore, the colluding peers always give

false feedback (*i.e.,* good feedback to malicious peers, and bad feedback to honest peers) making them easily detectable by a PSM algorithm. In a Sybil attack, some peers could act as moles by giving correct feedback for everyone in order to have a high credibility but giving false feedback about some malicious peers, in which case the PSM metric may not be successful.

Peertrust tackles turnover by having all feedback stored in a replicated DHT. Feedback is stored in r different peers in the DHT. When the feedback is retrieved all r peers return it. This replication serves two purposes: to stop malicious peers from falsifying the information in the DHT, and to be robust in high turnover environments.

Peers in Peertrust form a DHT where feedback is stored and retrieved from, thus there is a cost in trust assessments. The cost for a query or a storage operation in a regular DHT is $O(\log_2 n)$. In order to prevent malicious peers from removing feedback from the DHT, the data is replicated in different nodes. Therefore, the actual cost per DHT operation becomes $O(r \cdot \log_2 n)$, where r is the replication factor. If the designer wanted to make feedback removal impossible, he would have to set $r = n$. In this case there would be no need for peers to query the DHT for feedback, since all peers would have a copy of all feedback, but a storage operation would have a cost of $O(n)$. Furthermore, setting such a high value of replication would become a burden to peers, which would store too much data. Finally, each trust assessment requires $p + 1$ feedback requests, where p is the number of partners of the peer being assessed, thus the cost per assessment is $O((p + 1) \cdot r \cdot \log_2 n)$ when $r < n$ and $O(1)$ when $r = n$. Summing up, the cost per interaction includes: the number of assessments of potential targets t; the request and acknowledgement messages; and the feedback storage operations (two in the worst case). When $r < n$, this cost is $O((t+1) \cdot (p+1) \cdot r \cdot \log_2 n + 2 \cdot r \cdot \log_2 n + 2)$ which becomes $O(r \cdot n^2 \cdot \log_2 n)$ when the potential partners and previous partners are the whole set of peers. When $r = n$ the cost in messages is $O(n)$.

5.5.6 Repage

Repage [Sabater-Mir et al., 2006] is a sophisticated mechanism that uses cognitive science to model other agents. The main difference with other approaches is that it uses a meta-reasoning model in which image and reputation are separated. According to cognitive science, image is what an agent truly believes about another, based on its own experiences and the experiences of others, and reputation is a meta-model based on what the society as a whole thinks about a specific agent. The other models described so far in this section do not make this distinction, thus not allowing for a contradiction between image and reputation. Another difference is that the reputation of an agent is presented as a fuzzy set, as opposed to a single value as in other approaches. This allows the agent to have richer models of others, *e.g.,* 2/3 of the time it is a neutral partner and 1/3 of the times it is a good partner. Nonetheless, the author's effort in providing a sophisticated cognitive science model based on what humans do has the inconvenient of not taking into account the threats that are endemic

to electronic societies, such as whitewashing and Sybil attacks, which are made possible when identifiers are free, something that does not happen in human societies.

Badmouthing is tackled satisfactorily because Repage makes separate assessment on agents as interacting partners and as informants. An assessment of a partner that uses information from informants, weights this information according to the assessment of the reliability of these informants. This makes badmouthing attacks fruitless.

The Repage architecture does not provide a mechanism to avoid ballot-stuffing attacks. Agents are allowed to share whatever information they like. An agent's image as an informant can be degraded by issuing information that is not reliable. Therefore, ballot-stuffing could only be used to further improve the reputation and image of an agent with an already good reputation or image. This could be used to launch a dynamic personality attack. Nonetheless, repage tackles dynamic personality attacks by using a fuzzy set model of the peers. If the peer does many different actions this is reflected in the fuzzy set description.

Repage does not deal with the turnover problem since it has been designed for a multi-agent system in which all agents are assumed to be present all the time. Nonetheless, Repage can base its decisions solely on the information it has gathered in the past. Therefore, if some agents are no longer present, no new information can be gathered from them but the agent is still able to make its assessments. Furthermore, whitewashing and Sybil attacks are not dealt with, since the algorithm assumes that there exists a mechanism to prevent identity changes and false identity creation.

The work mentions that Repage is able to assess the reputation on a group of agents. This functionality assumes that the group being assessed is known beforehand. In the case of collusion, the colluding group will not advertise themselves. Therefore, it is hard for an agent to assess the reputation of a colluding group if it does not know which agents are part of it. The complexity of their algorithm is high in two ways. Firstly, the computational complexity of the algorithm that calculates the values for image and reputation is high since they use a fuzzy set approach with many values. Furthermore, Repage is integrated into the agent architecture as a model that gives complex information to the planner in order to help in the decision making. Secondly, the algorithm queries the agents in the system every time it wants to assess trust.

5.6 Proposed Reputation Mechanisms

This section presents two new reputation mechanisms that have been developed as part of this thesis to take the most advantage of the underlying contact network in our model. Specifically, they take into account the information given in the request routes as input to the reputation algorithm.

5.6.1 Route Enhanced Peertrust (REPT)

The reputation mechanisms defined in Section 5.5 are not robust against whitewashing attacks unless they use some sort of centralised identification scheme. Nonetheless, robustness against whitewashing can be achieved in a totally distributed manner by taking advantage of request route information. In this section we show how the Personal Similarity Measure (PSM) in Peertrust [Xiong et al., 2004] (see Section 5.5.5) can be made robust to whitewashing by using that information.

In our approach we modify the personalised similarity metric (PSM) of Peertrust so that it takes into account information about the request route to tackle whitewashing attacks.[3] We have used this metric because it has been shown to work well under badmouthing, ballot-stuffing, collusion, and dynamic personality attacks [Xiong et al., 2004]. Furthermore, Peertrust does not base its robustness on assumptions that need not hold (see Section 5.5.5).

Given an environment e, let $P(a,e) = \{a' \in A_e \mid \exists \langle a_s, a_r, \langle j, a_i, a_t, p\rangle\rangle \in \alpha(e)\ ((a_i = a \wedge a_t = a') \vee (a_i = a' \wedge a_t = a))\}$ be the set of partners of a *i.e.,* the agents that have sent an acknowledgement to a or received one from a (A_e is the set of agent identifiers present in the environment e, $\alpha(e)$ is the set of acknowledgements in e, a_i is the initiator and a_t is the target), $CP(a,a',e) = P(a,e) \cap P(a',e)$ denote the common set of partners of a and a', and $S(a,a',e)$ be the ratio between the sum of feedback values in feedback from a about a' and the number of acknowledgements where a and a' are partners.[4]

The PSM metric adapted to our scenario, without taking whitewashing into account, is shown below. The first parameter denotes the peer assessing the trust, the second parameter denotes the peer whose trust is being assessed, and the third is the environment over which the assessment is taken. When the denominator is 0, T_{PSM} cannot be calculated.

$$T_{PSM}(a,a',e) = \frac{\sum_{a_p \in P(a',e)} S(a_p,a',e) \times Sim(a,a_p,e)}{\sum_{a_p \in P(a',e)} Sim(a,a_p,e)}$$

Sim is defined as the root-mean-square similarity between the complaint rate vectors of the two peers:

$$Sim(a,a',e) = 1 - \sqrt{\frac{\sum_{a_p \in CP(a,a',e)} \Big(S(a,a_p,e) - S(a',a_p,e)\Big)^2}{|CP(a,a',e)|}}$$

The original Peertrust PSM is further modified to handle whitewashing by using feedback from interactions whose requests had been forwarded by any

[3] These modifications could be applied to other trust algorithms too.

[4] If there are no acknowledgements, $S(a,a',e) = 1$.

of the forwarding routers of the current request. This approximates the trust in the initiator by using information of previous interactions routed by those routers that trust the initiator. This variant is used when T_{PSM} cannot be calculated. Given an environment, let $RP(a, e)$ denote the set of routed partners in interactions where the request was forwarded by a; let $RS(a, e)$ denote the routed satisfaction, *i.e.,* the ratio of the sum of feedback values in feedback where the request was forwarded by a to the number of acknowledgements where the request was forwarded by a; let $CP(a, A', e)$ denote the common partners, *i.e.,* intersection between the set of partners of a and the set of partners of any peer in A'; and let $S'(A', a, e)$ denote the ratio of the sum of feedback values in feedback where any peer in A' was a partner of a to the number of acknowledgements where any peer in A' was a partner of a.[5]

The following equation describes the trust assessment metric adapted to handle whitewashing. The first parameter denotes the peer assessing the trust, and the second parameter is the set of routers that have trusted the assessed peer. If the denominator is 0, T_{WW} returns 1.

$$T_{WW}(a, A_r, e) = \frac{\sum_{a_r \in A_r} RS(a_r, e) \times Sim'(a, RP(a_r, e), e)}{\sum_{a_r \in A_r} Sim'(a, RP(a_r, e), e)}$$

Sim' is defined as the root-mean-square similarity between the complaint rates vector of the first peer and the accumulated complaint rates vectors of the peers in the second set:

$$Sim'(a, A', e) = 1 - \sqrt{\frac{\sum_{a_p \in CP(a, A', e)} \Big(S(a, a_p, e) - S'(A', a_p, e) \Big)^2}{|CP(a, A', e)|}}$$

Algorithm 2 shows how the router assesses whether to block a request or not. Notice that the whitewash-handling modification is only calculated when there is insufficient data to calculate the original metric. Furthermore, the metric is calculated in both directions: from the point of view of the request target assessing the initiator and from the point of view of the initiator assessing the target. Both assessments are combined to assess the satisfaction probability of the request for both agents.

In this approach, the agents only use their local environment to make assessments. Therefore, the only cost is that associated to the interaction protocol illocutions. The cost for routing requests through a network depends on both the routing mechanism and the network topology. Nonetheless, in the worst case scenario the cost of sending the request is $O(n)$ where n is the number of peers in the network. According to the behavioural properties in Section 5.3.1, since acknowledgements and complaints are only

[5] If there are no acknowledgements, $RS(r) = 1$ and $S'(W, p) = 1$.

Algorithm 2: The trust assessment algorithm for routers

Input: $\langle j, a_i, a_t, \langle\langle a_1, a_2, ..., a_n\rangle, A_b, A_r\rangle\rangle$
Input: $e \in E$
Output: assessed satisfaction probability
if $T_{PSM}(a_t, a_i, e)$ *can be calculated* **then**
    $t_i \leftarrow T_{PSM}(a_t, a_i, e)$;
else
    $A_f \leftarrow \{a_i | 1 < i \leq n\}$;
    $A_f \leftarrow A_f \setminus (A_b \cup A_r)$;
    $t_i \leftarrow T_{WW}(a_t, A_f, e)$;
end
if $T_{PSM}(a_1, a_t)$ *can be calculated* **then**
    $t_t \leftarrow T_{PSM}(a_i, a_t, e)$;
else
    $t_t \leftarrow T_{WW}(a_i, \{a_n\}, e)$;
end
return $t_i \cdot t_t$;

sent to the recipients in the request route, the complexity remains $O(n)$. Nonetheless, scale-free graphs have been shown to have a diameter $d = \ln \ln n$ [Bollobás and Riordan, 2004], and small-world graphs a diameter $d = \ln n$ [Nguyen and Martel, 2005, Martel and Nguyen, 2004]. Given that social networks are believed to possess both small-world and scale-free properties, we can assume that, given a good routing algorithm, the complexity of our model will be $O(\ln n)$.

5.6.2 Sybilproof Routing Mechanism (SRM)

In order to develop a Sybilproof routing mechanism we have chosen to satisfy all the requirements in Theorem 5.4.1. This section shows an algorithm that does this. In this algorithm the definition of an edge value must satisfy the restriction on edge splitting by which the values of the new edges must add up to the original edge.

In our model, the value that best matches this restriction for an edge going from agent a to agent a' when assessed by agent a_a is the following: let F' be the set of feedback illocutions sent by the a_a where a appears in the request route, the edge value is the ratio of the sum of feedback values in those feedback illocutions in F' where a forwarded the request to a' to the sum of feedback values in the feedback illocutions in F'. Let the function $\nu : A \times A \times A \times E \rightarrow \mathbb{R}$ represent the value of the edge going from the first agent to the second agent, when assessed by the third agent in a given environment, defined mathematically

as

$$\nu(a', a'', a_a, e) = \frac{\sum_{v' \in V'} v'}{\sum_{v'' \in V''} v''}$$

where $V' = \{v \mid \exists \langle \langle j, a_i, a_t, p \rangle, a_a, v \rangle \in e|_{a'}$ (a' had forwarded the request to a'')$\}$ and $V'' = \{v \mid \exists \langle \langle j, a_i, a_t, p \rangle, a_a, v \rangle \in e|_{a'}$ (a' had forwarded the request)$\}$. This edge value allows edge splitting. It is Sybilproof since it satisfies the restriction that all resulting edge values add up to the original edge value. By splitting an edge, the request (and thus the feedback) will travel through only one of the created edges.

The reputation mechanisms described as Sybilproof in Theorem 5.4.1 calculate the reputation from a set of paths from the calculating agent to the agent whose reputation is being calculated. In the interaction protocol proposed in Section 5.2 there is no such aggregation of paths, since the interaction request only needs to find one path p from the initiator to the target. In order for our mechanism to be Sybilproof as defined in Theorem 5.4.1, the path of forwarders that the request travels towards the target must be the one with the maximum g value. The router selection, when forwarding, must guarantee this. In order to guarantee this the router being selected must maximise the g value of the path out of the available ones. If there is a tie the one closest to the target will be chosen.[6]

There are many functions g that satisfy the requirements to become part of a Sybilproof reputation mechanism. We have chosen *min* because it is shown to be robust to simple collusion (or ballot-stuffing) in [Feldman et al., 2004a] as part of the maxflow algorithm. In that work they also provide a modification of the maxflow algorithm that is robust to badmouthing (*i.e.,* giving bad feedback) in which closer entities are believed more than farther entities. They can do this in their system because badmouthing can be detected as a conflict in feedback, whereas in our case feedback is totally subjective and cannot be verified.[7] Furthermore, we cannot use a g function that gives higher weights to edges that are closer to the target, since we would not be able to guarantee that the routing algorithm would find the maximum path, since the routing agent does not know how far away from the target it is.

As in REPT (see Section 5.6.1) agents using the SRM algorithm only take as input the local environment of the agent. Therefore, the cost asociated to SRM is that of the illocutions sent due to the interaction protocol. This allows us to prove that the cost is $O(\ln n)$.

[6]The shortest path heuristic reduces the number of illocutions being sent when compared to a random choice. Nonetheless, one could use other heuristics.

[7]although there could be a conflict in the feedback illocutions sent by the same agent to different recipients, we do not take this possibility into account for our work.

5.7 Analytical Comparison of mechanisms

The previous sections have shown how some of the most influential work in the reputation research area tackles the different security issues that can hinder the precision of a reputation algorithm. In this section we summarise the results from the comparison. Table 5.1 presents whether each algorithm described tackles each of the issues, where BM stands for badmouthing, BS for ballot-stuffing, Col for collusion, DP for dynamic personality, WW for whitewashing, Syb for Sybil, TO for turnover, Ml for mole and Cost for itself. The last column (Sec) provides the number of the section where the mechanism has been described.

In the case of the cells for the different adversarial attacks, Y means that the algorithm is robust to the attack, and N means that it is not. Some cells have special markers beside the Y which we now explain. Y* means that the robustness is based on the assumption that an agent with good reputation as a partner will return correct feedback, which need not always hold. Y† bases its robustness on the assumption that there are a set of peers that can always be trusted to give correct feedback, be them the oldest peers in the network or the ones with the highest reputation. This assumption also need not hold. Y‡ bases its robustness to whitewashing in giving new agents the lowest possible reputation, in which case no one will ever interact with them incurring in a bootstrap problem. At most they allow a certain random probability of interacting with an agent with the worst possible reputation, which makes the algorithm not robust to whitewashing. N• means that it is not robust since the algorithm assumes that there exists a mechanism to guarantee identification of agents, thus not allowing identity change or creation of multiple identities. Finally, N/A means that the algorithm could not be analysed for that specific property.

Table 5.1: Summary of the attributes for the different reputation algorithms

	BM	BS	Col	DP	WW	Syb	TO	Ml	Cost	Sec
YS	N	N	N	Y	N	N	Y	N	N/A	5.5.1
AD	Y*	N	N	N	N	N	Y	N	$O(w \log n)$	5.5.2
ET	Y†	Y	Y†	N	Y‡	N	Y	Y	$O(rn \log n)$	5.5.3
RD	Y	Y	N	Y	Y	N	Y	N	$O(rn \log n)$	5.5.4
PT	Y	Y	Y	Y	N	N	Y	N/A	$O(rn \log n)$	5.5.5
RPG	Y	N	N	Y	N•	N•	N	N/A	N/A	5.5.6
REPT	Y	Y	Y	Y	Y	N	Y	Y	$O(\ln n)$	5.6.1
SRM	Y	Y	Y	Y	Y	Y	Y	Y	$O(\ln n)$	5.6.2

5.8 Agent decisions

Agents make decisions during the interaction protocol. The decisions they make are the following: choosing a target for a request, choosing a certifying peer to which to forward a request, deciding whether to forward, block, discard or re-route a request, acknowledging a request, giving feedback for an interaction or not, and the feedback value for an interaction.

The model only observes the actions of agents, although it is reasonable to assume that agents decide according to the past history of events. In which case agents would be better off using a reputation mechanism which is robust against the different attacks defined in Section 5.4. What we do in this section is to define the properties that the agent decision making machinery must satisfy so that there is no change in the robustness of the reputation mechanism provided. Furthermore, we describe the consequences of the different decisions of an agent in our framework.

Choosing the target. The first decision that is made in an interaction protocol is choosing the target agent. The initiator selects a target agent from the set of certified agents. Although we make no assumption about how this decision is made, we do know that the agent cannot use the REPT and SRM reputation mechanisms, since at that point the agent does have route information. The initiator agent could use the PSM metric alone, but it would not be robust against Sybil attacks.

Routing the request. When dealing with the routing decision, if SRM is being used by the agents, there is a property to be satisfied: the maximum value path selection. As shown in Section 5.4.7 the path of forwarders in the request route that reached its target must be the one with the maximum value in order for the reputation mechanism to be Sybilproof.

Definition 5.8.1. Given a set of agent identifiers A, a set of content identifiers I, an environment e, an agent with identifier $a \in A$, and a function for calculating the value of a path g, a behaviour ι is said to be *route maximising* for agent a, if all the interaction requests it forwards are forwarded to an available agent that maximises the route value. *i.e.,* $(\iota(a, a', e) = \langle a, a', \langle j, a_i, a_t, \langle\langle a_1, a_2, ..., a_n\rangle, b, rr\rangle\rangle\rangle \wedge a_n \notin b \cup rr) \rightarrow (\forall a_p \in A_e \ (a \in A_{e|_{a_p}} \wedge g(\langle\langle a_1, a_2, ..., a_n\rangle, b, rr\rangle\rangle, a_t, e) \geq g(\langle\langle a_1, a_2, ..., a_p\rangle, b, rr\rangle\rangle, a_t, e)))$.

An agent can have a route maximising behaviour by always forwarding to the neighbour agent through which the target agent has gained most satisfaction. But this can be very inefficient from a routing point of view, since routing the request through an edge which does not have the highest value does not reduce the value of the path if a previous edge has an even lower value. By adding information of the edge values in the request route, an agent would have more options when choosing the agent to which to forward, which would allow an increase in routing efficiency. This information could be easily added to a

request route. Tampering would be avoided by the same encryption techniques described in Section 5.4.1. This information has not been added to the request routes in the experiments.

If an agent's behaviour is not route maximising, then the reputation mechanism is not proven to be Sybilproof. Nonetheless, the mechanism can still be *value Sybilproof* [Cheng and Friedman, 2005]. This means that a Sybil strategy would not be able to increase a Sybil node's reputation, but it would be able to decrease another agent's reputation. A malicious agent may have an incentive to do this if it was competing against the request initiator to interact with the request target. Imagine that Andrew wants to sell a car to Beatrix and Claude is also a car salesman. If Andrew manages to send a request to Beatrix and she acknowledges it, chances are that Claude will not be able to sell the car to Beatrix. If the request happens to be routed through Claude, he can use a non-route maximising behaviour so that Andrew's reputation is reduced, thus hoping that the request will not be acknowledged.

There is a side-effect to this behaviour that is not route-maximising, which is detrimental to Claude. If Claude manages to get Andrew's request discarded by Beatrix, and upon sending his own request it also gets discarded, then there is no satisfaction added to any of the edges coming out from Claude. Therefore, Claude's reputation remains lower than it would have, had Andrew's request been acknowledged and had the subsequent interaction satisfied Beatrix. It only pays for Claude to use a non route maximising behaviour if its current reputation is higher than Andrew's reduced one. Nonetheless, such information is not available to Claude, which removes the incentive for a behaviour that is not route maximising.

The next agent decision is what to do with a request: forward, block, reroute, or discard. We describe some properties associated to this decision: fraud avoiding, free-riding, and coercive.

Fraud avoiding. A behaviour associated to an agent avoids fraud if it discards requests that are not correct. Furthermore, it does not take into account acknowledgements that are not justified, feedback that is not legitimate, and any illocution that has been tampered with or not signed correctly. In other words, upon receiving any of the fraudulent illocutions above, the agent aborts the interaction protocol.

Agents executing non-fraud-avoiding behaviour only harm themselves, since they will be the only ones taking into account fraudulent illocutions. If an agent routes an interaction request that is not correct, the next agent can verify that the request is not correct, and it will discard the request if its behaviour is fraud-avoiding. A possible incentive for an agent to execute a fraud-avoiding behaviour is to collude with other agents in order to spread fraudulent illocutions. Nonetheless, the only agents they would manage to fool are those not presenting a fraud-avoiding behaviour.

Free-riding. Another behavioural property for an agent is free-riding. In general, an agent is said to free-ride when it consumes more than its fair share of a public resource, or shoulders less than a fair share of the costs of its production. In our scenario the public resource is the illocution request routing system. We do not tackle the issue of free-riding by sending too many requests, but we do tackle the issue of not forwarding the requests of other agents. For the first case we direct the reader to currency-based routing systems that tackle this issue [Pirzada et al., 2004]. Our model tackles the case in which an agent discards requests that are correct, have not been tampered with, and have been signed correctly, and it also tackles when it re-routes interaction requests which it could have forwarded. Notice that a behaviour that blocks a request does not make it free-riding. Blocking is part of the coercive process, as is giving feedback, and will be treated below.

As mentioned earlier, the main motivation for free-riding is to avoid the cost of routing interaction requests. When free-riding by re-routing, the agent is saving up bandwidth in case the request is hard to route, *i.e.,* the attempts to forward the request are not fruitful, thus increasing the number of interaction request illocutions to be sent. When free-riding via discarding, the savings are greater since not even the re-routed request is sent. Nonetheless, both cases of free-riding behaviour have negative consequences to the agent exhibiting it.

Let us assume that the reputation metric described in Section 5.6.2 is being used by the target of the request received by the router agent. In such case, an agent's reputation according to the target depends not only on the feedback about it, but also on the feedback of the target agent on interactions for which it had routed the request. Therefore, by free-riding, the router agent is reducing the opportunities through which it can improve its own reputation.

Coercive. The last property we define in the context of routing decisions is the coercive property. An agent's behaviour is coercive when it blocks some interaction requests.

We do not consider blocking to be a free-riding behaviour, since it is the essential part of the coercive process through which agents can punish others. By blocking a request, the blocking agent is making explicit its distrust of an agent in the request route, or the request route as a whole. Blocking was already mentioned in Chapter 4 as an enforcement mechanism for norms. In this chapter there are no global norms, so blocking becomes a coercive mechanism against other agents. Blocking can change the reputation of the agent. Since the routing mechanism chooses the highest valued path so far, blocking a request immediately makes the path value 0 and another path has to be found, which will probably have a lower reputation value than the current one could have. Furthermore, as in Chapter 4, if enough agents block a request it will not be able to get delivered. If such blocking behaviour persists an agent that is unpopular will be ostracised.

A blocking behaviour also has side-effects for the agent exhibiting it. Blocking a request has the same side-effects as re-routing it from the reputation point

of view. By blocking a request the agent is reducing its chances of increasing its own reputation. The difference with a free-riding behaviour through re-routing is the explicit expression of distrust.

Giving feedback. Finally, we describe the behavioural properties that deal with how agents send feedback. An agent that always sends feedback of its interactions to all agents in the request route has an informing behaviour. On the opposite end, an agent that never sends feedback about its interactions has a quiet behaviour.

Definition 5.8.2. Given an environment e, and an agent with identifier a, a behaviour ι is said to be *informing* for agent a, if there exist any legitimate feedback from a about another agent. *i.e.,* $\exists\langle a', a'', f\rangle \in e$ $(f = \langle\langle j, a_i, a_t, \langle\langle a_1, a_2, ..., a_n\rangle, b, rr\rangle\rangle, a, v\rangle \wedge \textbf{legitimate}(f))$.

Definition 5.8.3. Given an environment e, and an agent with identifier a, a behaviour ι is said to be *quiet* for agent a, if no legitimate feedback from a about another agent are present in the environment. *i.e.,* $\forall\langle a', a'', f\rangle \in e$ $(f = \langle\langle j, a_i, a_t, \langle\langle a_1, a_2, ..., a_n\rangle, b, rr\rangle\rangle, a_f, v\rangle \wedge \textbf{legitimate}(f) \wedge (a = a_i \vee a = a_t) \rightarrow a \neq a_f)$.

Assuming that the reputation mechanism described in Section 5.6.2 is being used by the agents in the system, agents have an incentive to give feedback about their interactions. By giving feedback an agent is raising the reputation not only of the agent that interacted with it, but also the reputation of all the agents that will be routed through any of the forwarding agents of the interaction request for which it is giving feedback. Since the reputation mechanism is subjective (meaning that the reputation of an agent depends on the feedback previously given by the assessing agent), the mechanism will be more precise the more feedback has been previously sent by the assessing agent. Furthermore, if an agent does not send feedback to the agents forwarding the interaction request, they cannot guarantee that future requests will be routed through a reputation maximising route.

On the other hand, by sending feedback the agent can also "annoy" the partner agent. Even if the feedback is not sent directly to the partner agent, the feedback can be forwarded to it. Given the fraud detection mechanisms explained in Section 5.4, the partner agent can be sure that the feedback is legitimate and not fraudulent. Therefore, a negative feedback can trigger retaliation by the agent that the feedback is mentioning. Such retaliation can come either by discarding, re-routing, or blocking future requests coming to or from the annoying agent, and by badmouthing the annoying agent in the future. Discarding and re-routing would fall into the free-riding problem, which has its own consequences, some of which have been discussed in this section. Blocking would be a legitimate coercive measure, since the trust of the agent giving the feedback has been reduced in the eyes of the agent the feedback is mentioning. Finally, badmouthing would only be possible if future interactions between these two agents took place again. Such interaction is unlikely to happen given that

both agents would have to agree to interact, specially if either of the agents has an incompatible avoiding behaviour (see Definition 5.3.5).

Having a quiet behaviour also has side effects. An agent benefits from the feedback of others, but since the agent already knows his subjective valuation of an interaction, sending out feedback would incur an extra cost to the agent. This type of problem has been termed "the tragedy of the commons" [Hardin, 1968, Hardin, 1998] and it could also be classified as free-riding. In this case, feedback would be the common good, which has to be maintained by all. If no one gives feedback, the reputation mechanism has no information on which to base its assessment. Luckily, the reputation mechanism in Section 5.6.2 establishes an incentive for agents to give feedback. If the agent does not send feedback to others, the route being chosen for future interaction requests may not be the one maximising the reputation. Therefore, the reputation mechanism is not Sybilproof for the agent having a quiet behaviour. This only affects the agent which has the quiet behaviour, thus the incentive can prevent the tragedy of the commons from happening, *i.e.,* not having enough feedback to make accurate reputation assessments.

5.9 Experiments

In this section we present the experiments that have been executed on the new MAN model. The experiments consist on many different simulations where the different enforcement mechanisms have been tested against the attacks defined in Section 5.4. The attacks can be combined, and there are simulations in which all the possible attack combinations have been tested.

In a simulation round each agent is given a chance to start an interaction protocol by being the initiator (in the case of a ballot-stuffing attack, the cheater is allowed to start five interaction protocols per round). The target selection is a stochastic decision based on the assessed trust for each peer, selecting with higher probability those peers with higher assessed trust. If the peer with the highest assessed trust is always selected, it is hard for newcomers to build up trust since they will hardly ever be selected as partners. Most reputation mechanisms tested do not use routing information. In those cases the requests are sent directly to the target, as if the network topology was complete. For the REPT and SRM mechanisms, peers make a stochastic decision on whether to block the request based on the assessed trust. A higher assessed trust makes for lower chances of being blocked. Finally, once the request reaches its target, the target peer makes another stochastic decision on whether to accept it or not based on its assessed trust in the initiator and request route.

In order to easily calculate peer satisfaction, we have defined an interaction as a game in which peers can either *collaborate* or *cheat*. Non-malicious peers will always collaborate, whereas malicious peers cheat some of the times. The amount of cheating done by malicious agents depends on the base cheating strategy they have chosen. Non-malicious peers want their partners to collaborate and are satisfied when that happens. When they are not satisfied they

complain. Malicious peers do not always complain when they are cheated on. In some sophisticated attack schemes they collude with other malicious agents in order to boost each others reputation (see Section 5.4 for the description of the adversarial attacks).

Name	**Type**
Agents	$2, \infty$
Rounds	$1, \infty$
Topology	Small World, Scale Free, Tree, Ring, Random,...
Cheater	$[0, 1]$
- Badmouthing	$\bot, \top$
- Ballot-Stuffing	$\bot, \top$
- Base Cheating	Always Cheat, Cheat Randomly, Dynamic Personality,...
- Whitewashing	$\bot, \top$
- Colluding	$\bot, \top$
Collaborator	$[0, 1]$
- Reputation Mechanism	YS, ET, PT, cPT[8] REPT, SRM,...

Table 5.2: Simulation Variables

Table 5.2 shows the variables in the simulation scenario. The *Base Cheating* variables define the way a malicious agent selects its action in a joint action when there is no other type of fraud mechanism in place, *e.g.,* badmouthing, ballot-stuffing, whitewashing, or colluding. The variables referring to the fraud mechanisms indicate whether the cheating agents will be using the attack to subvert the reputation mechanism in place. The other variables are common to the experiments in Chapter 4 and will not be discussed again.

Name	**Type**
Round	$\mathbb{N}$
Number of Messages (msg)	$\mathbb{N}$
Number of Joint Actions (ja)	$\mathbb{N}$
Number of Complaints (cmp)	$\mathbb{N}$
Number of Attempted Joint Actions (att)	$\mathbb{N}$
Number of False Positives (fp)	$\mathbb{N}$

Table 5.3: Simulation Measurements

Table 5.3 shows the measurements taken at each experiment. These measurements are taken at intervals of 10 rounds. All of these measurements have been described in Section 3.2 except for the false positives, which has been added to the simulation measurements in this scenario. A false positive is a term used in medical diagnosis to refer to the case in which a patient that does

not have the disease is diagnosed as having the disease. In our scenario a false positive occurs when the interaction attempt does not end in an interaction (*i.e.,* the reputation mechanism diagnoses that the interaction request will not bring about a satisfactory joint action). Nonetheless, had the joint action taken place it would have been satisfactory (*e.g.,* the diagnosis was flawed). In order to measure the false positives, the simulation tool runs the joint action for all requests that are either blocked or not acknowledged and if it is satisfactory it counts it as a false positive. The agents in the joint action choose their action but the joint action is not added to their memory, thus it is not used to update reputation values.

These measurements are used to calculate three metrics through which the reputation mechanisms are compared: Δ number of messages, Δ false negative rate, and Δ false positive rate, where Δ refers to the incremental nature of the metrics, *i.e.,* the metric only takes into account the events that happened since the previous measurement. As an example, the value for Δmessages at round 50 would be the measurement of the number of messages taken at round 50 minus the measurement taken at round 40. A false negative is another medical term that refers to the case in which a patient is not diagnosed with a disease when it is present. In the current scenario it equates to a complaint, *i.e.,* the reputation mechanism did not block the request and a joint action took place and was not satisfactory. The false negative rate (FNR), described in Equation 5.1 below, is used to measure how sensitive the treatment is (sensitivity is 1 - FNR). A high FNR implies a high probability that a non-satisfactory joint action attempt will be allowed to end up in a joint action. The false positive rate (FPR), described in Equation 5.2 below, is used to measure how specific the treatment is (specificity is 1 - FPR). A high FPR implies a high probability that satisfactory joint action attempts will not be allowed to end up in a joint action. Therefore, an optimal enforcement mechanism ought to have low FPR and FNR. In order to quantify an overall efficiency of the enforcement mechanism, the error rate (ER) has been calculated. Equation 5.3 below shows how the error rate is calculated. Both false positives and false negatives are considered errors, since they represent an undesirable outcome. Therefore, the error rate is the ratio of the total amount of errors over the total amount of instances, *i.e.,* interaction attempts.

$$FNR = \frac{\text{false negatives}}{\text{positive instances}} = \frac{cmp}{cmp + att - ja - fp} \tag{5.1}$$

$$FPR = \frac{\text{false positives}}{\text{negative instances}} = \frac{fp}{fp + ja - cmp} \tag{5.2}$$

$$ER = \frac{\text{false positives} + \text{false negatives}}{\text{instances}} = \frac{fp + cmp}{att} \tag{5.3}$$

The first set of experiments in this chapter are meant to test the reputation mechanism performance based on the different metrics previously described. The reputation mechanisms being tested are the ones described in Section 5.5 except for the following: Aberer and Despotovic, Reciprocative Decision, and Repage. The reason for not testing the first two is that the time they took to

evaluate each agent in the simulations was far too high even when the simulations were run with very few agents. Finally, Repage does not fit our scheme, since it uses gossip of reputation information and our model only allows direct interaction feedback to be shared.

The second set of experiments test how the other variables in the simulations affect the new mechanisms proposed in this dissertation, *i.e.,* REPT and SRM. The variables are the number of agents in the system, the number of rounds, the percentage of violator agents, and the different adversarial attacks.

5.9.1 Comparing mechanisms

This section presents the scenarios under which the different reputation algorithms have been tested, and the results form the experiments. Two scenarios have been used for the experiments: simple attack, and subversive attack. A simple attack is one where the malicious agents do not try to subvert the reputation algorithm in any way. That is, they do not perform badmouthing, ballot-stuffing, dynamic personality, whitewashing, collusion, or Sybil attacks. Table 5.4 shows the simple attack experiment design. On the other hand, a subversive attack is one in which the malicious agents try to subvert the reputation mechanism through any attack combination. Table 5.5 shows the subversive attack experiment design.

Name	**Values**
Agents	16, 32, 64, 128
Rounds	10, 20, ..., 100
Topology	Small World, Scale Free
Cheater	0.125, 0.25
- Badmouthing	$\perp$
- Ballot-Stuffing	$\perp$
- Base Cheating	Always, Randomly
- Whitewashing	$\perp$
- Colluding	$\perp$
Collaborator	0.75, 0.875
- Reputation Mechanism	YS, ET, PT, cPT, REPT, SRM
Name	**Type**
Messages	$\mathbb{N}$
Interactions	$\mathbb{N}$
Complaints	$\mathbb{N}$
Interactions blocked	$\mathbb{N}$
False Negatives	$\mathbb{N}$
Integrity Constraints	**Formula**
Full Partitioning	Cheater + Collaborator = 1.0

Table 5.4: Simple attack experiment design table

Name	**Values**
Agents	16, 32, 64, 128
Rounds	10, 20, ..., 100
Topology	Small World, Scale Free
Cheater	0.125, 0.25, 0.5, 0.75, 0.875
- Badmouthing	$\bot, \top$
- Ballot-Stuffing	$\bot, \top$
- Base Cheating	Always, Randomly, Dynamic
- Whitewashing	$\bot, \top$
- Colluding	$\bot, \top$
Collaborator	0.125, 0.25, 0.5, 0.75, 0.875
- Reputation Mechanism	YS, ET, PT, cPT, REPT, SRM
Name	**Type**
Messages	$\mathbb{N}$
Interactions	$\mathbb{N}$
Complaints	$\mathbb{N}$
Interactions blocked	$\mathbb{N}$
False Negatives	$\mathbb{N}$
Integrity Constraints	**Formula**
Full Partitioning	Cheater + Collaborator = 1.0

Table 5.5: Subversive attack experiment design table

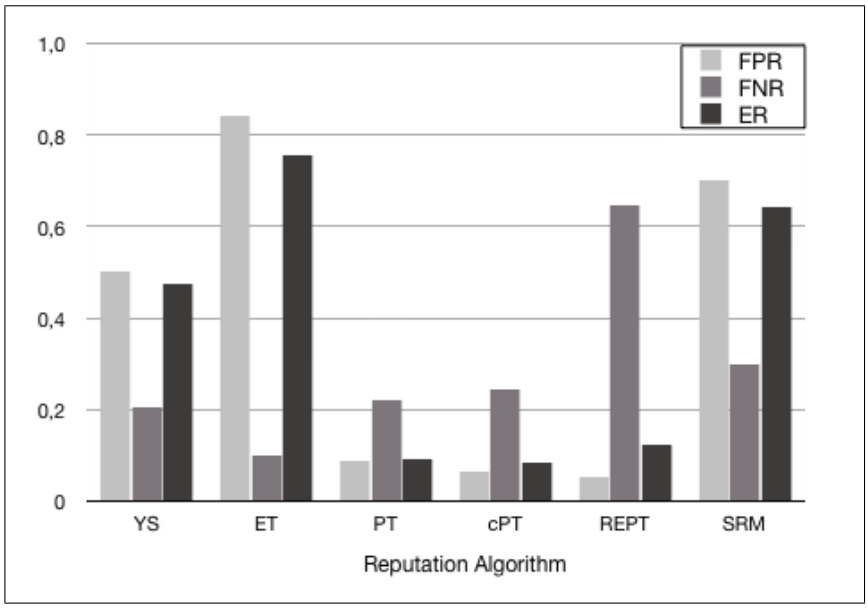

Figure 5.3: Comparison of the different reputation mechanisms for simple attacks.

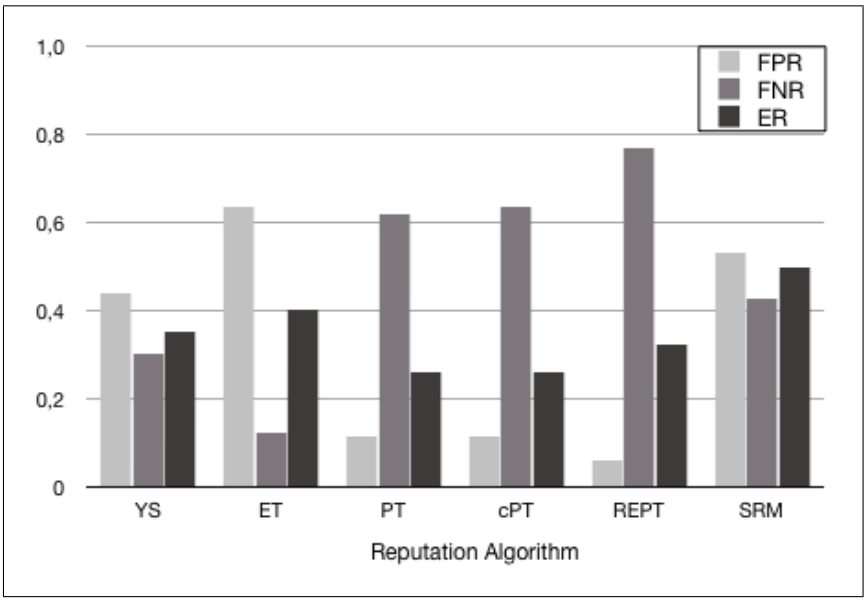

Figure 5.4: Comparison of the different reputation mechanisms for subversive attacks.

The ANOVA tests showed that the choice of reputation mechanism was statistically significant for all the outcome metrics (FNR, FPR, ER, and messages), for both experiments.

Figure 5.3 presents a graph with the different rates for each of the reputation mechanisms returned in the simulations under the simple attack scenario. The Peertrust algorithms are not significantly different to one another in the FNR, FPR, and ER results, according to the Tukey tests. These algorithms have the lowest average ER, although they do not have the lowest FPR or FNR. REPT has the largest FNR, and the lowest FPR, with the second lowest average ER. The rest of the algorithms have a low FNR (Eigentrust having the lowest) but a rather high FPR, and ER. This is due to the fact that on average, there are more interaction protocols that would end in satisfactory joint actions than end in complaints. Therefore, false positives are a larger part of all the errors and influence the ER more heavily.

Figure 5.4 shows the rates for the subversive attack scenario. In this scenario it is also the case that the Peertrust algorithms perform the best for the error rates. Furthermore, REPT still has the lowest FPR and the largest FNR, which makes its ER larger than that of the Peertrust algorithms. The other algorithms seem to be in a similar ordering to the one for the simple attack scenario. Nonetheless, it is worth noting that the FPR and ER went down for YS, ET, and SRM when compared to the simple attack scenario. This could be due to the fact that colluding peers do not complain about one another, and there are less chances for false positives. Finally, the FNR went up for all

mechanisms, showing that the subversive attacks were successful to some degree in all reputation mechanisms..

The number of messages generated through each of the mechanisms had great disparities. Table 5.6 shows the average messages sent. YS, REPT, and SRM had the lowest number of messages, followed by ET, cPT, and finally PT, which had the largest number of messages. The Tukey tests for both scenarios showed that the number of messages for YS, REPT, and SRM was not significantly different. Furthermore, the Tukey test for the subversive attack scenario showed a significant difference among all the other reputation mechanism. On the other hand the Tukey test for the simple attack scenario only showed a significant difference for the regular Peertrust algorithm.

Mechanism	**Simple**	**Subversive**
YS	6.3×10^3	1.4×10^4
ET	1.0×10^6	2.0×10^6
PT	4.2×10^7	5.3×10^7
cPT	5.1×10^5	3.6×10^6
REPT	2.8×10^3	1.0×10^4
SRM	7.3×10^3	1.4×10^4

Table 5.6: Average number of messages by reputation mechanisms

5.9.2 Analysing REPT and SRM

This section shows how the different variable settings affect the performance of the reputation mechanisms proposed in this thesis: REPT and SRM. In order to do this, two full-factorial experiments have been run in a subversive scenario for each of the reputation mechanisms. Tables 5.7 and 5.8 present the experiment design for the REPT and SRM analysis, respectively. The analysis consists on seeing how each of the input variables in the scenario affects the performance metrics (FPR, FNR, ER, and number of messages).

The statistical analysis for the REPT experiment showed that the input variables were statistically significant in all cases, except for ballot-stuffing, which was not significant for the FNR outcome. As the number of agents increased over 32 so did the FNR and the ER, and the FPR went down. Between 16 and 32 agents the trend was inverted. Simulations with small-sorld networks had larger FNR, ER, and number of messages, but lower FPR than those simulations with scale-free networks. Furthermore, as the percentage of violators increased so did the FNR, FPR, ER, and the number of messages. Nonetheless, the FNR and ER peaked at some point between 50% and 75% of violators. Finally, as the number of rounds increased, the FNR and ER became smaller up until round 50 and 30, respectively, where the values stabilised. On the other hand, the FPR increased up until round 30 and then stabilised, and the number of messages was not affected by the number of rounds.

Name	**Values**
Agents	16, 32, 64, 128
Rounds	10, 20, ..., 100
Topology	Small World, Scale Free
Cheater	0.125, 0.25, 0.5, 0.75, 0.875
- Badmouthing	$\bot, \top$
- Ballot-Stuffing	$\bot, \top$
- Base Cheating	Always, Randomly, Dynamic
- Whitewashing	$\bot, \top$
- Colluding	$\bot, \top$
Collaborator	0.125, 0.25, 0.5, 0.75, 0.875
- Reputation Mechanism	REPT
Name	**Type**
Messages	$\mathbb{N}$
Interactions	$\mathbb{N}$
Complaints	$\mathbb{N}$
Interactions blocked	$\mathbb{N}$
False Negatives	$\mathbb{N}$
Integrity Constraints	**Formula**
Full Partitioning	Cheater + Collaborator = 1.0

Table 5.7: REPT analysis experiment design table

Name	**Values**
Agents	16, 32, 64, 128
Rounds	10, 20, ..., 100
Topology	Small World, Scale Free
Cheater	0.125, 0.25, 0.5, 0.75, 0.875
- Badmouthing	$\bot, \top$
- Ballot-Stuffing	$\bot, \top$
- Base Cheating	Always, Randomly, Dynamic
- Whitewashing	$\bot, \top$
- Colluding	$\bot, \top$
Collaborator	0.125, 0.25, 0.5, 0.75, 0.875
- Reputation Mechanism	SRM
Name	**Type**
Messages	$\mathbb{N}$
Interactions	$\mathbb{N}$
Complaints	$\mathbb{N}$
Interactions blocked	$\mathbb{N}$
False Negatives	$\mathbb{N}$
Integrity Constraints	**Formula**
Full Partitioning	Cheater + Collaborator = 1.0

Table 5.8: SRM analysis experiment design table

Table 5.9 shows how the different subversive attacks influenced the performance of the reputation mechanism as quantified by the mean difference in the false negative, false positive, and error rates. Even though the mechanism was design to counteract whitewashing attacks, it was this type of attack that modified its performance the most. Although its performance was not always lowered. REPT had an interesting performance against badmouthing attacks: the FNR and FPR were decreased and yet the overall ER was increased. Ballot-stuffing did not influence the FNR, but it did increase the FPR and ER. Dynamic personality attacks were effective at raising the FPR and FNR, yet the overall ER was lowered. Finally, REPT worked better under collusion attacks, for which all the metrics were decreased.

Attack	**FNR**	**FPR**	**ER**
Badmouthing	-10.4	-1.3	10.9
Ballot-stuffing	$\sim$	1.4	8.9
Dynamic Personality	8.9	2.7	-2.0
Whitewashing	34.1	-8.4	15.2
Collusion	-0.4	-6.4	-28.0

Table 5.9: Results on attacks to REPT. Mean difference in percentage between simulations with and without the attack.

The statistical analysis for the experiments on the SRM mechanism, showed that all input variables were statistically significant for all the measured metrics. The number of agents influenced the FNR by reducing it up to 32 agents and increasing it after 64. It influenced the FPR by increasing it up to 64 agent and reducing it afterwards. It influenced the ER by increasing it up to 64 agents, after which it stayed the same. Finally, the number of messages was always increased.

The network topology was also shown to modify the values of the outcome metrics. The small-world networks had lover FNR and ER, whereas they had higher FPR and number of messages. Higher percentage of violator agents brought about higher FNR and number of messages, and lower FPR and ER.

The number of rounds also affected the metrics by decreasing the FNR, which got stabilised after round 70, increasing the FPR, which got stabilised after round 60, increasing the ER, which got stabilised after round 40, and increasing the number of messages up to round 20, after which it was lowered, but not significantly.

Table 5.10 presents the influence of the different subversive attacks on the performance of SRM. This influence is quantified by the mean of the difference between the simulations in which malicious agents did not attempt the attack and their respective counterparts where they did. SRM's performance was not modified much by dynamic personality and badmouthing attacks. Its performance improved under collusion and ballot-stuffing attacks. Whereas whitewashing attacks triggered much higher FNR, much lower FPR, and slightly higher overall error rates (ER).

Attack	FNR	FPR	ER
Badmouthing	0.43	-0.39	-4.2
Ballot-stuffing	-10.1	-1.7	-6.1
Dynamic Personality	-0.69	-5.3	0.71
Whitewashing	33.1	-21.7	3.3
Collusion	-9.7	-16.1	-13.7

Table 5.10: Results on attacks to SRM. Mean difference in percentage between simulations with and without the attack.

5.10 Applications

The model and techniques in this chapter are general enough to be applicable to most distributed applications (*e.g.,* P2P, MAS, GRID) with some modifications. The main modification is to create an underlying social network through which all interactions will happen. This sections presents two applications to which the enforcement techniques in this chapter can be applied.

The first application is LiquidPub: a distributed scientific publishing application that aims to change the way scientific knowledge is produced, disseminated, evaluated, and consumed [Giunchiglia et al., 2009]. In LiquidPub there are three main entities: the scientific knowledge object (SKO) where the scientific information is stored, the researchers which create, modify, and maintain the SKOs, and finally an ontology of research fields. The ontology is structured as a tree, but the SKOs and the researchers are structured as networks (For a detailed description of the LiquidPub framework go to http://liquidpub.org). The users in LiquidPub adopting the role of researchers can realise many actions such as creating, modifying, or deleting an SKO. Nonetheless, it is the joint actions between researchers that interest us, since it is through these that the enforcement techniques in this chapter can be applied. Researchers have a subjective definition on which joint actions are satisfactory (*e.g.,* an author may disapprove of a reviewer's comments on its paper). Furthermore, researchers can use complex adversarial techniques to subvert the reputation algorithms used to evaluate their SKOs. These characteristics make LiquidPub an ideal candidate for the enforcement techniques in this chapter.

The second application is a P2P messaging system. There are many implementations for this type of application in the market. The functionality provided by most of them allows users to chat with one another, independently of their network situation and configuration. These messaging systems also provide tools to manage contact lists, which makes for an almost seamless integration with the model defined in this chapter.

5.10.1 LiquidPub

This section presents how the enforcement techniques in this chapter can be embedded into the LiquidPub system. The fist thing to be addressed is to find

appropriate matches between the entities in LiquidPub and the entities in the model described in Section 5.1.

Each researcher in the LiquidPub system is assigned an agent identifier, thus becoming the counterpart of an agent in the model. SKOs and the ontology do not have counterparts in the model. The identity certifications do not have a direct counterpart in LiquidPub, since it does not address free identifiers in its framework. Researchers in LiquidPub are assumed to have been given an identifier by an authority which guarantees that the researcher exists. In order to remove this restriction we can add identity certifications either explicitly by having researchers certify each other's identity, or implicitly through one of its joint actions. The easiest way to do this is to implicitly certify any researchers identity with which one has co-authored a paper. Certification cancellations, on the other hand, would have to be explicit, since there is no action in LiquidPub through which it could be implied satisfactorily.

The co-authoring joint action is not the only joint action that can be executed by researchers in LiquidPub. Researchers can also execute the following joint actions: submit a paper to a conference, review or comment a paper, select programme committee members for a conference, and sending call for papers for a conference. In order to execute any of the previous joint actions, one of the involved researchers would have to act as the initiator of the interaction protocol and the other researcher would be the target. For example, if a conference charter wants to select a researcher as part of its programme committee, the researcher in charge of the charter would send a request for interaction to the researcher, if the requests reaches its target and the researcher acknowledges it, then the joint action can take place. By acknowledging the request the researcher does not necessarily accept to be in the programme committee. The negotiation for that would take place in the joint action, where one of the possible outcomes is that the researcher becomes part of the conference's programme committee. After the joint action takes place, both the initiator and the target may give feedback about the joint action. If the programme committee member takes too long to fulfil its responsibilities, the initiator would give negative feedback about the joint action. On the other hand, if the programme committee member feels he had a low work load it would give positive feedback about the joint action.

The modelling of researcher reputation is a very important part of LiquidPub, specially for credit attribution. The reputation algorithms designed for LiquidPub are used mainly to evaluate the quality of a scientific knowledge object. Nonetheless, there are also mechanisms designed to evaluate the reputation of researchers, although it is mainly as the creator of quality SKOs. Therefore, the reputation mechanisms present in LiquidPub, or others presented in this chapter, can be used in the interaction protocol as part of the enforcement mechanism. By using the enforcement techniques in this chapter the satisfaction rate in joint interactions would increase, even if the researchers used the adversarial techniques presented in Section 5.4 to subvert the reputation mechanisms.

5.10.2 P2P Messaging

There are a number of applications that use P2P technology in order to get users to communicate through the internet. Some of the best known are Skype and MSN Messenger. These P2P networks allow users to chat with one another either by using text messages, voice conversation, or video conference. This section shows how the enforcement mechanisms in this chapter can be used in these applications.

In P2P messaging applications, most of the times conversations are held by users that are direct contacts of one another. Nonetheless, it is possible to start a conversation with a user that is not on one's contact list. Conference calls in which not all users are contacts of one another are perfect examples of this. Furthermore, a request for becoming a contact can be sent to any user in the system. P2P messaging applications have more functionality than this. Nonetheless, for the sake of the example at hand, only the conversation and contact request functionality will be taken into account. Both of these events are mapped to joint actions in the model. When the joint action is a contact request which is executed correctly, an identity certification is added for each of the users. This functionality, already present in these applications, forms the contact network as specified in the model.

The software in these applications allows a user to specify whether or not she wants to allow other users not on her contact list to start a conversation with her. This measure in itself is a type of distributed enforcement. The enforcement mechanisms presented in this chapter can also be used in such types of networks, by having the joint actions (conversations and contact requests) embedded in the interaction protocol described in this chapter. This would mean that the user's software would have to send a request through the user's contacts for every conversation and contact request. Furthermore, the user would be allowed to complain about joint actions which she did not consider satisfactory. The enforcement mechanisms presented in this chapter could be applied by the users in order to reduce the amount of undesired behaviour (*e.g.,* spam) to which they are exposed.

5.11 Discussion

This chapter defines a model for peer enforcement that can be used in situations where there is too much uncertainty for the model in Chapter 4 to be applicable. These situations include scenarios in which the network topology is dynamic, when there is open access for new agents with free identifiers, where the modelling of a relationship between agents is a directed one, where the definition of a satisfactory interaction is subjective and potentially different for each agent, or where the malicious agents exhibit complex behaviours in order to evade punishment. Therefore, the model and techniques presented in this chapter are suitable to many more real-life applications. Two of the potential applications have been described in Section 5.10.

The description of the model in this chapter is more detailed, since it is intended to be applicable for networked applications. Therefore, special consideration has been taken in describing the means by which encryption techniques can be used to prevent data fraud. As in Chapter 4, there is an analysis of the amount of complaints that can be achieved. Nonetheless, there is a problem with the results in that they give upper bounds on the number of legitimate complaints based on the number of certified agents. This made perfect sense in the previous chapter, where the agents were not allowed to change identities, nor where new agents allowed into the system. Since that is no longer the case, the results from Section 5.3 are not very insightful.

Given that the applications for which the model is intended are full of uncertainty, the mechanisms used to model the behaviour of other agents need to take this into account. Some of the most influential reputation algorithms have been analysed and implemented in order to compare them to two reputation mechanisms that have been designed to take advantage of the MAN model in this chapter. Out of the reputation algorithms those that were shown to be most suitable have been implemented and compared in experiments with the two new ones (REPT and SRM). For the experiments, a simulation tool has been implemented that allows the researcher to run simulations with different input parameters. The experiments presented here test the performance of the reputation algorithms under all input variable combinations. Including most of the well-known adversarial attacks that malicious agents can attempt in order to subvert the reputation mechanism: badmouthing, ballot-stuffing, dynamic personality, whitewashing, and collusion.

The main concern in this research is to find enforcement mechanisms that are both robust and scalable. A mechanism is robust when the adversarial attacks do not reduce its performance. And it is scalable when its cost becomes too great when the number of agents increases. Both REPT and SRM have been shown to be more scalable than most of the other mechanisms. Only one of the other mechanisms had an equivalent scalability. On the other hand the performance when measures through the different error rates (false positive rate, false negative rate, and overall error rate) was not the best of all of them. This we expected because the information available to the most scalable mechanisms is less than that available to the least scalable ones.

A scalability issue that we have not dealt with in this chapter is the router selection algorithm. Since the identifier space is flat, existing routing algorithms may not scale well. Maintaining a routing table becomes cumbersome in large environments. In order to minimise the impact of routing tables, compact routing schemes [Thorup and Zwick, 2001] can be used. This is possible because social networks have scale-free and small-world properties. Current approaches to compact routing assume that a full view of the network is available to all. Although this can easily be achieved by setting up a DHT where each peer's contacts are stored, it could be possible to rely on local knowledge from past routed interactions in order to improve routing efficiency. Nonetheless, such compact schemes introduce a small increase in path length. But this is not a problem, since the main concern in our approach is not the efficiency of routing

but the increase in satisfaction. In this case longer paths may be beneficial since interaction feedback is spread to more peers.

The robustness of the algorithms varies depending on the attack scenario and the metric used to measure the robustness. The REPT mechanism was shown to have the lowest FPR, but its FNR was the highest. In situations where it is inadmissible to block interactions that are valid (such as with spam control in the P2P messaging application) using REPT would be the best choice. On the other hand, if the application user deems it best to make sure that its interactions are satisfactory even at the expense of blocking some potentially satisfactory interactions, then the best choice would not be REPT but Eigentrust, although one would need to take into account that Eigentrust is not as scalable, and that its overall error rate is high. The SRM mechanism has been shown to be mediocre at best. Its performance as measured in error rates is low. On the bright side the effect from adversarial attacks is not as high. Probably due to the fact that the performance is bad to start with.

The enforcement mechanisms available from the model in this chapter have the following benefits: they are totally distributed, relatively easy to implement, they allow self-policing (peers can have different definitions of satisfactory interactions, and different thresholds for punishing those that deviate), and they are scalable when compared to other approaches.

The robustness against malicious activities of the reputation mechanisms proposed in this chapter ought to be improved. Furthermore, more enforcement mechanism are possible under this model which have not been studied. Such as sanctioning routers for not doing a good job at enforcing behaviour, or modifying the agent's contact links as a way to position it in an area of the network with a common understanding of what a satisfactory interaction is. Furthermore, the fact that a router agent has blocked a request could be used as feedback to be taken into account by the reputation mechanisms. Finally, another interesting area of research is to use MANET trust based routing techniques to tackle the free-riding problem present in many distributed applications where there is resource sharing.

Chapter 6

Conclusions

Enforcement is the act or process of compelling observance of a kind of behaviour. The work developed in this thesis has addressed the issue of enforcement in distributed artificial environments. These artificial environments have been modelled in a way that is understandable to humans, thus, the electronic agents in these systems can be expected to possess some traits that are common to humans in human societies. Gregariousness is one of the traits that is reflected in artificial environments. We have developed enforcement mechanisms that are based on the assumption that artificial agents are gregarious. Even though there is no physical space in electronic environments, there is a virtual space in which electronic agents come together to interact with others in order to achieve their personal goals.

Electronic agents in multiagent systems often need to interact with others. An interacting agent expects the outcome of an interaction to be satisfactory for it. Otherwise it would not interact with others, because co-ordination has associated costs. Sometimes, in order for an interaction to be satisfactory for an agent, the outcome of the interaction may not be satisfactory to the other agents involved in the interaction. Agents will try to avoid these situations through what we describe as enforcement techniques. Enforcement techniques are applied by agents in order to increase the probability of being satisfied with the outcome of interactions they participate in. The work in this thesis has studied different enforcement mechanisms that agents can apply in order to increase the satisfaction probability.

The techniques provided attempt to thwart the ability to interact of those agents for which enforcement is intended. In order for agents to have power over the capability of others to interact, there must be dependencies between agents that have to be satisfied during the interaction process. In this thesis, social relationships have been exploited for this purpose by including them in the interaction protocol. First, the multiagent system has been structured by setting up a social network. Secondly, interaction protocols have been defined that allow agents execute joint actions with one another. The use of the social relationships between agents is a key design characteristic in these inter-

action protocols. Therefore, dependencies among agents have been forced into the interaction process, thus, allowing enforcement techniques based on these dependencies to be embedded into the interaction protocols.

The enforcement techniques have been studied under two scenarios. The first scenario is a normative multiagent system in which the definition of a satisfactory interaction is given beforehand and shared by all the agents in the system. This is a closed system in which the social network is fixed, *i.e.,* no new agents are allowed, and the social structure of the system is static. The restrictions in this scenario limit the types of applications for which the enforcement mechanisms are suitable. Large-scale electronic systems have been designed in order to be dynamic, both allowing new agents to join, and existing agents to modify their network relationships. Furthermore, these systems do not impose global norms that dictate the appropriate behaviour. Each agent has a subjective definition of what a satisfactory interaction is, and this definition is not shared with others explicitly. The second scenario where the enforcement techniques have been studied is suited to those types of applications in which the restrictions in the first scenario do not hold. Studying how the enforcement techniques behave in these two scenarios, which are so different, one gets a broad understanding of the applicability in different types of distributed applications.

The thesis has described some of the applications that can benefit from the enforcement techniques in both scenarios. Distributed information sharing fora where the rules about information sharing are known to all those in the group, and self-repairing sensor networks designed for a single application both fit into the the restrictive scenario of Chapter 4. Therefore, the designer of such applications would be able to define upper bounds on the number of norm violations or sensor failures. On the other hand, P2P messaging applications, and LiquidPub (distributed application for scientific publishing) do not fit into the restrictive scenario because there is a degree of subjectivity as to what is a satisfactory behaviour, and because the social structure is highly dynamic. The designers of these types of applications would benefit from the work in Chapter 5, by embedding into their applications the enforcement mechanisms and reputation mechanisms that are robust, scalable, and efficient as their needs be.

In environments where enforcement techniques are used, intelligent cheating agents (*i.e.,* those agents that get satisfaction from exhibiting behaviours that are not satisfactory to others) will try to avoid enforcement. Adversarial behaviours meant to avoid the enforcement mechanisms can vary in the degree of complexity. But their main goal is to change the perception of others so that the model that others make of the society is changed in order to benefit them. These types of attacks are mainly focused on subverting the reputation mechanism, which is how the agents model the probability of satisfaction, and on which they base the decision of sanctioning other agents through the enforcement mechanisms. In the static scenario only simple adversarial behaviours have been taken into account. Those which can be expected from the agents in the types of applications to which the scenario is applicable. In the uncertain scenario the adversarial behaviours can be very complex, including attacks such

as badmouthing, ballot-stuffing, whitewashing, dynamic personality, and collusion. In each of the scenarios being studied, the enforcement mechanisms have been observed under the relevant adversarial attacks.

In order to study the behaviour of the multiagent networks (MAN) analytically, we have defined a mathematical model for each of the scenarios. Through these models we have calculated analytical results which gave us upper bounds on the number of unsatisfactory joint actions that could be executed against either any agent of the system, or a subset of agents with certain behavioural properties. In the static scenario the upper bounds have been shown to depend on the number of agents and the enforcement techniques used. The enforcement techniques that have been studied in this scenario are:

- **Avoiding** — Not interacting with violator agents.
- **Blocking** — Not disclosing contacts to violator agents.
- **Protecting** — Not disclosing the contacts that are violators.
- **Informing** — Making public the joint actions where the partner violated the norm.

When an agent exhibits the avoiding enforcement behaviour, it will receive at most as many norm violations as agents in the system. Therefore, if a group of agents of size m all exhibit the avoiding behaviour, the upper bound on the number of norm violations they can receive is $m(n-1)$ where n is the total number of agents in the system. On the other hand if a connected group of agents A' exhibit the four enforcement behaviours it will at most a number of norm violations that is lower that the previous one (see Section 4.3.5 for an in depth discussion of the formulas defining the maximum number of norm violations). These results are important because they do not require that all the agents exhibit certain behaviours. Thus allowing a subset of the agents in a system to achieve a degree of enforcement which grants them a maximum number of violations to be received by them.

In the dynamic scenario some similar properties have been calculated analytically based on the enforcement techniques used. In this case the techniques are:

- **Networked** — Discarding those illocutions containing requests that are not correct, acknowledgements that are not justified, or feedback that is not legitimate.
- **Avoiding** — Not interacting with incompatible agents.
- **Blocking** — Not forwarding or re-routing requests with incompatible agents as partners.
- **Informing** — Giving feedback about joint actions.

When an agent exhibits the networked and avoiding enforcement behaviours, it will execute at most as many non-satisfactory joint actions as certified identities in the system. In a similar fashion one could demonstrate the upper bounds when all agents exhibited the four enforcement behaviours. However, there are two issues with these upper bounds. Firstly, they are based on the definition of incompatible agents. This definition is not straightforward in an uncertain environment where each agent has a subjective definition of what is a satisfactory joint action. Secondly, they are based on the number of certified identities. Making the upper bounds meaningless, since any agent can change its identity or create as many false identities as it pleases.

The analytical results were insufficient in the dynamic scenario. The static scenario did give some analytical results that were relevant. Notwithstanding, we were not able to prove many other insights we had about the workings of the enforcement mechanisms in both scenarios. In order to test these insights about how the new enforcement techniques would work when applied to real systems in both scenarios, an empirical perspective was used. Those insights could then be tested through experiments. For this purpose, we developed a simulation tool for each of the models defined in this thesis. These tools allow the researcher to simulate a society of agents where the models are in place and to set the values for the different model parameters. Factorial experiments were easily conducted as multiple instances of the simulation tool with different values.

Some of those insights we had were supported by the experiments. Firstly, the data from the simulations supported the hypotheses that stricter norm enforcement behaviours, larger ratios of agents exhibiting enforcement behaviours, and specific types of network typology all reduced the number of norm violations. Secondly, the simulation data also supported the hypothesis that using the request data from the proposed MAN interaction protocol to gather information about other peers allowed reputation mechanisms to reduce the number of messages. Finally, the simulation data supported the hypothesis that the information gathered from the request data could help in reducing some types of subversive attacks on the reputation mechanisms. These are all major contributions because they showed that the enforcement mechanisms devised in this thesis are useful, and that they could be applied in ways that are scalable and robust.

By exploring the large amounts of data generated throughout the simulations, new and surprising insights arouse about the behaviour of the system as a whole. Sometimes even debunking the original insights that we had about the system. For example, tree networks turned out to be worse at lowering norm violations and random networks did best. This was specially surprising to us because in previous models without any type of adversarial behaviour the tree network had performed best. Another example of these surprising new insights is that the informing enforcement behaviour does not bring about a large increment in satisfaction to agents. This could mean that it is not that important for agents to exhibit an informing behaviour in order to achieve norm enforcement. This is specially interesting since getting users to give feedback is a difficult

issue for most applications were there is a reputation algorithm in place.

Although we have made an effort to describe a model in which the enforcement techniques can be applied that is comprehensive, there are still some requirements that the model builds on that have to be satisfied by other technologies that have not been discussed. Electronic applications that seek to apply the enforcement techniques from this thesis must implement the following technologies: public key cryptography, a scalable flat-space-identifier routing protocol, and a protocol that manages illocution transportation across different network configurations.

These technologies have issues that will have to be tackled by the application developers. Firstly, although public key cryptography is commonly used in many distributed applications such as email certification, however, to the best of our knowledge, there is no large-scale system that makes use of this type of cryptography to the same extent as needed in order to avoid data fraud in our model. Therefore, there might be a scalability issue from having to sign multiple times each illocution if the cryptographic package is not fast at encrypting and decrypting. Secondly, There are many P2P large-scale applications that have solved the information transportation across networks almost seamlessly by using proxies and other technologies, nevertheless, open source solutions to these problems are packaged as part of other software and they may be hard to re-use. Finally, existing technologies for flat-space-identifier routing protocols are either not scalable or are based on the hypothesis that all routing agents have complete knowledge of the network topology. This pre-requisite is hard to implement in large-scale systems while maintaining scalability. Specially if the network topology is highly dynamic. In order to reap the benefits in scalability of the interaction protocol defined in Chapter 5 incremental solutions for flat-space-identifier routing will have to be developed.

Throughout the thesis the underlying motivation has been to give tools to developers of distributed applications that would allow the users of their applications to take an active part in building a community in which they could have satisfactory interactions. This is achieved by giving users some power over other users that would allow them to enforce socially acceptable behaviours. The reasons that led us to concentrate on distributed mechanisms without any trace of centralisation are two-fold. On one hand, it is an issue of scalability, since centralised components tend to be the bottle-necks as soon as the number of users grow. On the other hand, it is an issue of robustness. When you grant anyone, and everyone, the power to enforce through sanctions, there is always the fear that these powers may be corrupted. It is important to make it hard for this to happen. The enforcement techniques that have been studied avoid giving any subset of agents enough power that allows them to ruin another agent for their own profit. This has been tested through experiments as has the efficiency of the enforcement techniques. Another driving point in these techniques that differs from other enforcement research is to avoid retaliation by harming others. With the exception of informing others through gossip, all the actions that agents execute in order to enforce norms are based on non-cooperation as opposed to actions that directly harm those for which they are intended. The

work described in this thesis is a glimpse into what can be achieved by using this type of techniques. However, much more can be accomplished with this mindset.

6.1 Future work

Being an engineer at heart I believe that after the research advanced in this thesis, the next step ought to be in the direction of technology transfer by implementing a framework for the development of applications that takes advantage of the enforcement techniques described in this thesis. The framework should be built mixing concepts from P2P, MAS, and Web Services, in order to allow framework users to develop using both an agent-oriented development style or the more common process interface metaphor. The framework would use P2P communication techniques that have proven to work for large-scale systems and allow users behind all sorts of network configurations to join the application. Furthermore, the framework would provide the tools for contact management and identity management, for encryption and decryption techniques to avoid data fraud, for interaction protocol management, and it would sanction management tools either through the implementation of some reputation mechanisms or by allowing the user to decide the sanctions directly. By developing this framework we would allow other researchers to pursue the research lines that have been opened in this thesis.

Were the framework to be used by many types of applications, it would be a waste of time having to maintain different social networks for each application. In order to allow different applications to work on the same framework and social network simultaneously, the interaction protocol should be modified in order to allow information about the purpose of the joint action to be included into the requests. This opens a new line of research where many new aspects of enforcement can be studied. Some of these aspects are: rich models of users which take into account the different applications the framework is being used for; cross-domain enforcement techniques where actions in one application can trigger sanctions in another application (*e.g.,* not fulfilling an obligation from an auction application might not only get your requests for that application blocked in the future, but also the requests to create new joint events in a calendar application); finally, ontology alignment can be studied by having the content in the new fields in the request and feedback illocutions be defined through terms of an ontology.

Another interesting functionality to implement in the framework is to allow multi-user joint actions. Examples of applications where this would be needed are chats, forums, and auctions. Having multi-user joint actions opens another research line, which consists on developing community enforcement techniques designed specifically for multi-user scenarios. In such cases, the interaction request would be sent to a community or group of agents which would have previously defined the process through which to decide whether to acknowledge the request once it got there. The coupling of enforcement techniques with

coalition theory and social choice theory is a long-term research line. As an example of where this research line might start, let us imagine that in a specific community it is the user that starts the auction that defines the rules that govern the process through which new users join the auction. One possibility is to have the new user send a request to the auction creator which would be in charge of granting access to the new user. Another possibility is to have the new user send requests to each of the current members of the group and access would be granted to the group either by majority vote or by unanimity. In the last case, if any of the requests does not make it to one of the group members, the users would not be able to join. Through this scenario we can study new enforcement techniques that are tailored to the different procedures for joining a community and interacting as part of it.

One of the limitations of the approaches proposed in this thesis is that peers are assumed to co-operate in the interaction protocol by forwarding interaction requests and replying to contact queries. Therefore, an application using these protocols could be hampered because of free-riding. It is not clear whether the system could function under different levels of free-riding by agents that do not want to share the cost of maintaining the interaction protocol. Furthermore, a new user of the system, which would be connected by just a few or no contacts, may have bootstrap problems because of free-riding and turnover (users may not always be connected). This problem brings about a short-term research line through which techniques such as currency management and public access hubs can be studied. Through this research line we would study what effect the use of techniques to counteract free-riding would have on the different enforcement mechanisms provided in this thesis. Furthermore, we could also study the new types of attacks and the robustness and scalability of the interaction protocol when such techniques are used. A public access hub is a node that will become a contact of anyone that asks, which improves scalability and robustness of the interaction protocol, but reduces the effectiveness of enforcement techniques. Whereas a micro-payment scheme is a technique through which a currency is created in order to be used to pay for request routing. This solution has problems of its own, such as fraud and currency management. Furthermore, the changes that would have to be made to the protocol in order to use one of these schemes could have an impact on the robustness of the system, since new attacks could be possible.

Finally, another line of research has been opened in this thesis, namely the study of new enforcement mechanisms that cater to the dynamic model in Chapter 5. We mention below some of these enforcement mechanisms, which have not been studied as part of this thesis due to the difficulty of defining experiments in which to test them out under the current simulation testbed. Having a real world application that used this technology would help define such experiments, since it would provide a much richer testbed for the simulations. Firstly, a user could cancel contact relationships. This would have the same effect as permanently blocking all requests coming from a contact. The intended effect is to break ties with a complete portion of the network. Secondly, blocking requests from routers which do not block correctly. Through this enforcement

technique the user is sanctioning bad routing behaviour and thus tries to avoid collusion. Thirdly, not forwarding requests through the sanctioned agents. This is a technique whose purpose is twofold. On one hand, it would lower the reputation of the sanctioned user in the eyes of those using a reputation mechanism similar to SRM. On the other hand, if a micro-payment scheme is being used to prevent free-riding, this enforcement technique would lower the ability of the sanctioned user to get its requests delivered. Finally, adding router feedback to the interaction request route in order to increase the information available to the reputation algorithms would also force each router to define its trust on the request, which would make detection collusion easier.

The development of the techniques and improvements described in this section via future work, will help mature the field of distributed enforcement to a point were it is accessible to application developers. Furthermore, a framework as the one proposed would bring together two of the main fields in distributed systems that have the most to gain from one another: Peer-to-Peer and Multi-Agent Systems.

Bibliography

[Aberer et al., 2003] Aberer, K., Cudré-Mauroux, P., Datta, A., Despotovic, Z., Hauswirth, M., Punceva, M., and Schmidt, R. (2003). P-grid: a self-organizing structured p2p system. *SIGMOD Record*, 32(3):29–33.

[Aberer and Despotovic, 2001] Aberer, K. and Despotovic, Z. (2001). Managing trust in a peer-2-peer information system. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 310–317, New York, NY, USA. ACM Press.

[Ågotnes et al., 2007] Ågotnes, T., der Hoek, W. V., Rodríguez-Aguilar, J. A., Sierra, C., and Wooldridge, M. (2007). On the logic of normative systems. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 07)*, pages 1175–1180. AAAI Press.

[Aldewereld et al., 2006] Aldewereld, H., Dignum, F., García-Camino, A., Noriega, P., Rodríguez-Aguilar, J. A., and Sierra, C. (2006). Operationalisation of norms for usage in electronic institutions. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 223–225, New York, NY, USA. ACM.

[Axelrod, 1985] Axelrod, R. (1985). *The Evolution of Cooperation*. Basic Books.

[Axelrod, 1986] Axelrod, R. (1986). An evolutionary approach to norms. *The American Political Science Review*, 80:1095–1111.

[Barabasi and Albert, 1999] Barabasi, A. L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439):509–512.

[Boella and Lesmo, 2001] Boella, G. and Lesmo, L. (2001). Deliberate normative agents. In Conte, R. and Dellarocas, C., editors, *Social order in MAS*. Kluwer Academic Publishers.

[Boella and van der Torre, 2005] Boella, G. and van der Torre, L. W. N. (2005). Enforceable social laws. In *AAMAS*, pages 682–689.

[Bollobás and Riordan, 2004] Bollobás, B. and Riordan, O. (2004). The diameter of a scale-free random graph. *Combinatorica*, 24(1):5–34.

[Briggs and Cook, 1995] Briggs, W. and Cook, D. (1995). Flexible social laws. In Mellish, C., editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 688–693, San Francisco. Morgan Kaufmann.

[Broersen et al., 2001] Broersen, J., Dastani, M., Hulstijn, J., Huang, Z., and van der Torre, L. (2001). The BOID architecture: conflicts between beliefs, obligations, intentions and desires. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 9–16, New York, NY, USA. ACM Press.

[Broersen et al., 2004] Broersen, J., Dignum, F., Dignum, V., and Meyer, J.-J. C. (2004). Designing a deontic logic of deadlines. In *7th Int. Workshop of Deontic Logic in Computer Science (DEON'04)*, pages 43–56, Portugal.

[Cabri et al., 2006] Cabri, G., Ferrari, L., Leonardi, L., and Quitadamo, R. (2006). Collaboration-driven role suggestion for agents. In *Proceedings of IEEE-06 Workshop on Distributed Intelligent Systems*.

[Carpenter et al., 2004] Carpenter, J., Matthews, P., and Ong'ong'a, O. (2004). Why punish: Social reciprocity and the enforcement of prosocial norms. *Journal of Evolutionary Economics*, 14(4):407–429.

[Castelfranchi, 2000] Castelfranchi, C. (2000). Engineering social order. In *ESAW '00: Proceedings of the First International Workshop on Engineering Societies in the Agent World*, volume 1972, pages 1–18. Springer-Verlag.

[Castelfranchi et al., 1998] Castelfranchi, C., Conte, R., and Paoluccci, M. (1998). Normative reputation and the costs of compliance. *Journal of Artificial Societies and Social Simulation*, 1(3).

[Castelfranchi et al., 1999] Castelfranchi, C., Dignum, F., Jonker, C. M., and Treur, J. (1999). Deliberative normative agents: Principles and architecture. In *Agent Theories, Architectures, and Languages*, pages 364–378.

[Castelfranchi et al., 2003] Castelfranchi, C., Giardini, F., Lorini, E., and Tummolini, L. (2003). The prescriptive destiny of predictive attitudes: From expectations to norms via conventions. In Alterman, R. and Kirsh, D., editors, *XV Annual Conference of the Cognitive Science Society*.

[Cheng and Friedman, 2005] Cheng, A. and Friedman, E. (2005). Sybilproof reputation mechanisms. In *P2PECON '05: Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, pages 128–132, New York, NY, USA. ACM.

[Chisholm, 1963] Chisholm, R. M. (1963). Contrary-to-duty imperatives and deontic logic. *Analysis*, 24(2):33–36.

[Cholvy and Cuppens, 1995] Cholvy, L. and Cuppens, F. (1995). Solving normative conflicts by merging roles. In *Fifth International Conference on Artificial Intelligence and Law*, Washington, USA.

[Coen, 2000] Coen, M. H. (2000). Non-deterministic social laws. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 15–21. AAAI Press / The MIT Press.

[Conte and Castelfranchi, 1995] Conte, R. and Castelfranchi, C. (1995). Understanding the functions of norms in social groups through simulation. In Gilbert, N. and Conte, R., editors, *Artificial Societies: The Computer Simulation of Social Life*, chapter 13, pages 213–226. UCL Press.

[Cranefield, 2005] Cranefield, S. (2005). A Rule Language for Modelling and Monitoring Social Expectations in Multi-Agent Systems. Technical Report 2005/01, University of Otago.

[Cranefield, 2007] Cranefield, S. (2007). Modelling and monitoring social expectations in multi-agent systems. In *Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems II*. Springer-Verlag.

[Craven and Sergot, 2008] Craven, R. and Sergot, M. (2008). Agent strands in the action language nc+. *Journal of Applied Logic*, 6(2):172–191.

[Damiani et al., 2002] Damiani, E., di Vimercati, D. C., Paraboschi, S., Samarati, P., and Violante, F. (2002). A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 207–216, New York, NY, USA. ACM.

[Delgado, 2002] Delgado, J. (2002). Emergence of social conventions in complex networks. *Artificial Intelligence*, 141(1):171–185.

[Dignum et al., 2002] Dignum, F., Kinny, D., and Sonenberg, L. (2002). Motivational attitudes of agents: On desires, obligations, and norms. In *CEEMAS '01: Revised Papers from the Second International Workshop of Central and Eastern Europe on Multi-Agent Systems*, pages 83–92, London, UK. Springer-Verlag.

[Douceur, 2002] Douceur, J. R. (2002). The sybil attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251–260, London, UK. Springer-Verlag.

[Erdös and Rényi, 1960] Erdös, P. and Rényi, A. (1960). *On the evolution of random graphs*, volume 5, pages 17–61. Publ. Math. Inst. Hung. Acad. Science.

[Esteva et al., 2002] Esteva, M., Padget, J., and Sierra, C. (2002). Formalizing a Language for Institutions and Norms. In *ATAL '01: Revised Papers from the 8th International Workshop on Intelligent Agents VIII*, volume 2333 of *Lecture Notes in Artificial Intelligence*, pages 348–366. Springer-Verlag.

[Esteva et al., 2001] Esteva, M., Rodríguez-Aguilar, J.-A., Sierra, C., Garcia, P., and Arcos, J.-L. (2001). On the Formal Specification of Electronic Institutions. In *Agent-mediated Electronic Commerce (The European AgentLink Perspective)*, volume 1991 of *LNAI*. Springer-Verlag.

[Esteva et al., 2004] Esteva, M., Rosell, B., Rodriguez-Aguilar, J. A., and Arcos, J. L. (2004). AMELI: an agent-based middleware for electronic institutions. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, pages 236–243. IEEE Computer Society.

[Feldman et al., 2004a] Feldman, M., Lai, K., Stoica, I., and Chuang, J. (2004a). Robust incentive techniques for peer-to-peer networks. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 102–111, New York, NY, USA. ACM.

[Feldman et al., 2004b] Feldman, M., Papadimitriou, C., Chuang, J., and Stoica, I. (2004b). Free-riding and whitewashing in peer-to-peer systems. In *Proceedings of SIGCOMM'04 Workshop*. ACM.

[Fitoussi and Tennenholtz, 2000] Fitoussi, D. and Tennenholtz, M. (2000). Choosing social laws for multi-agent systems: minimality and simplicity. *Artificial Intelligence*, 119(1-2):61–101.

[Fon and Parisi, 2005] Fon, V. and Parisi, F. (2005). The behavioral foundations of retaliatory justice. *Journal of Bioeconomics*, 7(1):45–72.

[Fugger, 2009] Fugger, R. (2009). Ripple.

[Gaertner et al., 2007] Gaertner, D., García-Camino, A., Noriega, P., Rodríguez-Aguilar, J. A., and Vasconcelos, W. (2007). Distributed norm management in regulated multiagent systems. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, New York, NY, USA. ACM.

[Gaertner et al., 2006] Gaertner, D., Noriega, P., and Sierra, C. (2006). Extending the BDI architecture with commitments. In *Proceedings of the Ninth International Conference of the Catalan Association for Artificial Intelligence (CCIA' 2006)*.

[Gaertner et al., 2009] Gaertner, D., Rodríguez-Aguilar, J. A., and Toni, F. (2009). Agreeing on institutional goals for multi-agent societies. In *Coordination, Organizations, Institutions and Norms in Agent Systems IV: COIN 2008 International Workshops, COIN@AAMAS 2008*, pages 1–16, Berlin, Heidelberg. Springer-Verlag.

[Garcia and Hoepman, 2005] Garcia, F. D. and Hoepman, J.-H. (2005). Off-line karma: a decentralized currency for peer-to-peer and grid applications. *Lecture Notes in Computer Science*, 3531:364–377.

[Garcia-Camino et al., 2005] Garcia-Camino, A., Noriega, P., and Rodriguez-Aguilar, J. A. (2005). Implementing norms in electronic institutions. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 667–673, New York, NY, USA. ACM Press.

[García-Camino et al., 2006] García-Camino, A., Noriega, P., and Rodríguez-Aguilar, J.-A. (2006). An Algorithm for Conflict Resolution in Regulated Compound Activities. In *Seventh Annual International Workshop Engineering Societies in the Agents World (ESAW'06)*.

[García-Camino et al., 2007] García-Camino, A., Rodríguez-Aguilar, J.-A., Sierra, C., and Vasconcelos, W. (2007). Norm-Oriented Programming of Electronic Institutions: A Rule-based Approach. In *Coordination, Organizations, Institutions, and Norms in Agent Systems II*, volume 4386 of *Lecture Notes in Computer Science*, pages 177–193.

[García-Camino et al., 2006] García-Camino, A., Rodríguez-Aguilar, J. A., Sierra, C., and Vasconcelos, W. W. (2006). A rule-based approach to norm-oriented programming of electronic institutions. *Sigecom Exchanges*, 505:33–40.

[Giunchiglia et al., 2009] Giunchiglia, F., Chenu-Abente, R., Osman, N. Z., Sabater, J., Sierra, C., Xu, H., Babenko, D., and Schneider, L. (2009). Design of the sko structural model and evolution. Project Deliverable D1.2v1, Liquid Publications.

[Governatori, 2005] Governatori, G. (2005). Representing business contracts in *RuleML*. *International Journal of Cooperative Information Systems*, 14(2-3):181–216.

[Grizard et al., 2007] Grizard, A., Vercouter, L., Stratulat, T., and Muller, G. (2007). A peer-to-peer normative system to achieve social order. In *Coordination, Organizations, Institutions, and Norms in Agent Systems II*, volume 4386, pages 274–289. Springer-Verlag.

[Hales, 2002] Hales, D. (2002). Group reputation supports beneficent norms. *Journal of Artificial Societies and Social Simulation*, 5(4).

[Hardin, 1968] Hardin, G. (1968). The tragedy of the commons. *Science*, 162(3859):1243–1248.

[Hardin, 1998] Hardin, G. (1998). Extensions of "the tragety of the commons". *Science*, 280(5364):682–683.

[Hübner et al., 2006] Hübner, J. F., Sichman, J. S., and Boissier, O. (2006). S-$MOISE^+$: A middleware for developing organized multi-agent systems. In *Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems*, volume 3913 of *Lecture Notes in Computer Science*, pages 64–78. Springer-Verlag.

[Jackson, 2003] Jackson, M. O. (2003). Mechanism theory. In Devigs, U., editor, *Optimization and Operations Research*, The Encyclopedia of Life Support Science. EOLSS Publishers, Oxford, UK.

[Joseph et al., cess] Joseph, S., Sierra, C., Schorlemmer, M., and Dellunde, P. (Advanced Access). Deductive coherence and norm adoption. *Logic Journal of the IGPL*.

[Kaminka et al., 2002] Kaminka, G. A., Pynadath, D. V., and Tambe, M. (2002). Monitoring teams by overhearing: A multi-agent plan-recognition approach. *JAIR*, 17(83-135).

[Kamvar et al., 2003] Kamvar, S. D., Schlosser, M. T., and Garcia-Molina, H. (2003). The eigentrust algorithm for reputation management in p2p networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 640–651, New York, NY, USA. ACM Press.

[Kittock, 1994] Kittock, J. E. (1994). The impact of locality and authority on emergent conventions: initial observations. In *AAAI '94: Proceedings of the Twelfth National Conference on Artificial Intelligence*, volume 1, pages 420–425, Menlo Park, CA, USA. American Association for Artificial Intelligence.

[Kollingbaum and Norman, 2004] Kollingbaum, M. and Norman, T. (2004). Strategies for resolving norm conflict in practical reasoning. In *ECAI Workshop Coordination in Emergent Agent Societies 2004*.

[Kollingbaum and Norman, 2003a] Kollingbaum, M. J. and Norman, T. J. (2003a). NoA - a normative agent architecture. In *Proceedings of the International Joint Conference on Artificial Intelligence 2003*, pages 1465–1466.

[Kollingbaum and Norman, 2003b] Kollingbaum, M. J. and Norman, T. J. (2003b). Norm adoption in the NoA agent architecture. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 1038–1039, New York, NY, USA. ACM Press.

[Legras, 2002] Legras, F. (2002). Using overhearing for local group formation. In *Proceedings of AAMAS-02*.

[López y López and Luck, 2004] López y López, F. and Luck, M. (2004). Normative agent reasoning in dynamic societies. In *Proceedings of The Third International Joint Conference on Autonomous Agents and Multi Agent Systems AAMAS'04*.

[Loukos and Karatza, 2009] Loukos, F. and Karatza, H. D. (2009). Reputation based friend-to-friend networks. *Peer-to-Peer Networking and Applications*, 2(1):12–23.

[Marín and Sartor, 1999] Marín, R. H. and Sartor, G. (1999). Time and norms: a formalisation in the event-calculus. In *ICAIL '99: Proceedings of the 7th international conference on Artificial intelligence and law*, pages 90–99, New York, NY, USA. ACM Press.

[Martel and Nguyen, 2004] Martel, C. and Nguyen, V. (2004). Analyzing kleinberg's (and other) small-world models. In *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 179–188, New York, NY, USA. ACM.

[Marti et al., 2004] Marti, S., Ganesan, P., and Garcia-Molina, H. (2004). Sprout: P2p routing with social networks. *Lecture Notes in Computer Science*, Current Trends in Database Technology - EDBT 2004 Workshops:425–435.

[Maskin and Sjöström, 2002] Maskin, E. S. and Sjöström, T. (2002). Implementation theory. In Arrow, K. J., Sen, A. K., and Suzumura, K., editors, *Handbook of Social Choice Theory and Welfare*. North-Holland, Amsterdam.

[Meyer, 1988] Meyer, J.-J. C. (1988). A different approach to deontic logic: deontic logic viewed as a variant of dynamic logic. *Notre Dame journal of formal logic*, 29(1):109–136.

[Minsky, 1991a] Minsky, N. H. (1991a). The imposition of protocols over open distributed systems. *IEEE Transactions on Software Engineering*, 17(2):183–195.

[Minsky, 1991b] Minsky, N. H. (1991b). Law-governed systems. *Softw. Eng. J.*, 6(5):285–302.

[Minsky, 1991c] Minsky, N. H. (1991c). Law-governed systems. *Softw. Eng. J.*, 6(5):285–302.

[Minsky and Rozenshtein, 1988] Minsky, N. H. and Rozenshtein, D. (1988). A software development environment for law-governed systems. In *SDE 3: Proceedings of the third ACM SIGSOFT/SIGPLAN software engineering symposium on Practical software development environments*, pages 65–75, New York, NY, USA. ACM Press.

[Nguyen and Martel, 2005] Nguyen, V. and Martel, C. (2005). Analyzing and characterizing small-world graphs. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 311–320, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

[Nisan, 2007] Nisan, N. (2007). Introduction to mechanism design (for computer scientists). In Nisan, N., Roughgarden, T., Tardos, E., and Vazirani, V. V., editors, *Algorithmic Game Theory*. Cambridge University Press, Cambridge, UK.

[Novick and Ward, 1993] Novick, D. and Ward, K. (1993). Mutual beliefs of multiple conversants: A computational model of collaboration in air traffic control. In *Proceedings of AAAI-93*.

[Oliver, 1980] Oliver, P. (1980). Rewards and punishments as selective incentives for collective action: Theoretical investigations. *American Journal of Sociology*, 85(6):1356–75.

[Padget and Bradford, 1999] Padget, J. A. and Bradford, R. J. (1999). A pi-calculus model of a spanish fish market - preliminary report. In *AMET '98: Selected Papers from the First International Workshop on Agent Mediated Electronic Trading on Agent Mediated Electronic Commerce*, pages 166–188, London, UK. Springer-Verlag.

[Parkes, 2001] Parkes, D. (2001). *Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency*. PhD thesis, University of Pennsylvania, Department of Computer and Information Science.

[Perreau de Pinninck et al., 2008a] Perreau de Pinninck, A., Gutnik, G., and Kaminka, G. A. (2008a). Reducing communication cost via overhearing. In *Proceedings of the Sixth European Workshop on Multi-Agent Systems*.

[Perreau de Pinninck et al., 2007] Perreau de Pinninck, A., Sierra, C., and Schorlemmer, M. (2007). Friends no more: Norm enforcement in multi-agent systems. In *Proceedings of the sixth international joint conference on Autonomous Agents and Multi-Agent Systems*, pages 1–3. ACM.

[Perreau de Pinninck et al., 2008b] Perreau de Pinninck, A., Sierra, C., and Schorlemmer, M. (2008b). Distributed norm enforcement: Ostracism in multi-agent systems. In *Computable Models of the Law*, volume 4884 of *Lecture Notes in Artificial Intelligence*, pages 275–290. Springer-Verlag.

[Pirzada et al., 2004] Pirzada, A. A., Datta, A., and MacDonald, C. (2004). Trust-based routing for ad-hoc wireless networks. In *Proceedings of the 12th IEEE Interantional Conference on Newtorks*, volume 1, pages 326–330.

[Pujol et al., 2005] Pujol, J. M., Delgado, J., Sangüesa, R., and Flache, A. (2005). The role of clustering on the emergence of efficient social conventions. In *IJCAI '05: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 965–970.

[R-Project, 2009] R-Project (2009). R-project.

[Ramchurn et al., 2004] Ramchurn, S. D., Jennings, N. R., Sierra, C., and Godo, L. (2004). Devising a trust model for multi-agent interactions using confidence and reputation. *Applied Artificial Intelligence*, 18(9-10):833–852.

[Rao and Georgeff, 1995] Rao, A. S. and Georgeff, M. P. (1995). BDI agents: From theory to practice. In *Proceedings of the First International Conference on Multiagent Systems (ICMAS)*, pages 312–319, San Francisco, California, USA.

[Robertson, 2005] Robertson, D. (2005). A lightweight coordination calculus for agent systems. In *Declarative Agent Languages and Technologies II*, volume 3476, pages 183–197. Springer-Verlag.

[Rossi and Busetta, 2004] Rossi, S. and Busetta, P. (2004). Towards monitoring of group interactions and social roles via overhearing. In *Proceedings of CIA-04*.

[Rossi and Busetta, 2005] Rossi, S. and Busetta, P. (2005). With a little help from a friend: Applying overhearing to teamwork. In *Proceedings of IJCAI-05 Workshop on Modelling Others from Observations (MOO)*.

[Ryu and Lee, 1995] Ryu, Y. U. and Lee, R. M. (1995). Defeasible deontic reasoning and its applications to normative systems. *Decission Support Systems*, 14(1):59–73.

[Sabater and Sierra, 2001] Sabater, J. and Sierra, C. (2001). Regret: reputation in gregarious societies. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 194–195, New York, NY, USA. ACM.

[Sabater-Mir et al., 2006] Sabater-Mir, J., Paolucci, M., and Conte, R. (2006). Repage: Reputation and image among limited autonomous partners. *Journal of Artificial Societies and Social Simulation*, 9(2):3.

[Sartor, 1991] Sartor, G. (1991). Legal reasoning and normative conflicts. In *Legal Knowledge Based Systems : Model-based legal reasoning (JURIX 1991)*.

[Sartor, 1992] Sartor, G. (1992). Normative conflicts in legal reasoning. *Artificial Intelligence and Law*, 1(2-3):209–235.

[Savarimuthu et al., 2007] Savarimuthu, B. T., Purvis, M., Cranefield, S., and Purvis, M. (2007). Role model based mechanism for norm emergence in artificial agent societies. In *Proceedings of the International Workshop on Coordination, Organization, Institutions, and Norms (COIN), Honolulu, Hawai'i, USA*.

[Sergot, 2001] Sergot, M. (2001). A computational theory of normative positions. *ACM Transactions on Computational Logic*, 2(4):581–622.

[Shoham and Tennenholtz, 1995] Shoham, Y. and Tennenholtz, M. (1995). On Social Laws for Artificial Agent Societies: Off-line Design. *Artificial Intelligence*, 73(1-2):231–252.

[Sierra and Debenham, 2005] Sierra, C. and Debenham, J. (2005). An information-based model for trust. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 497–504, New York, NY, USA. ACM.

[Sun et al., 2005] Sun, L., Jiao, L., Wang, Y., Cheng, S., and Wang, W. (2005). An adaptive group-based reputation system in peer-to-peer networks. In *Proceedings of the 1st Workshop on Internet and Network Economics*, volume 3828, pages 651–659. Springer Berlin.

[Tan and van der Torre, 1996] Tan, Y.-H. and van der Torre, L. W. (1996). How to combine ordering and minimizing in a deontic logic based on preferences. In *Deontic Logic, Agency and Normative Systems, Proceedings of the Deltaeon'96 Workshops in Computing*, pages 216–232. Springer Verlag.

[Taylor, 1982] Taylor, M. (1982). *Community, Anarchy & Liberty*. Cambridge University Press.

[Tennenholtz, 1998] Tennenholtz, M. (1998). On stable social laws and qualitative equilibria. *Artificial Intelligence*, 102(1):1–20.

[Thorup and Zwick, 2001] Thorup, M. and Zwick, U. (2001). Compact routing schemes. In *SPAA '01: Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures*, pages 1–10, New York, NY, USA. ACM.

[Tian et al., 2008] Tian, Y., Wu, D., and Ng, K.-W. (2008). On distributed rating systems for peer-to-peer networks. *Computer Journal*, 51(2):162–180.

[Upadrashta, 2005] Upadrashta, Y. (2005). *Social Routing*. PhD thesis, University of Saskatchewan.

[Vasconcelos et al., 2007] Vasconcelos, W., Kollingbaum, M. J., and Norman, T. J. (2007). Resolving conflict and inconsistency in norm-regulated virtual organizations. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*.

[Vazquez-Salceda et al., 2004] Vazquez-Salceda, J., Aldewereld, H., and Dignum, F. (2004). Implementing norms in multiagent systems. In Lindemann, G., Denzinger, J., Timm, I., and Unland, R., editors, *Multiagent System Technologies*, LNAI 3187, pages 313–327. Springer-Verlag.

[Vishnumurthy et al., 2003] Vishnumurthy, V., Chandrakumar, S., and Sirer, E. G. (2003). Karma: A secure economic framework for peer-to-peer resource sharing. In *Workshop of the Economics of Peer-to-Peer Systems*.

[von Ahn et al., 2003] von Ahn, L., Blum, M., Hopper, N. J., and Langford, J. (2003). CAPTCHA: Using hard AI problems for security. In *Advances in Cryptology – EUROCRYPT 2003*, volume 2656/2003 of *Lecture Notes in Computer Science*, page 646. Springer Berlin.

[von Wright, 1951] von Wright, G. H. (1951). Deontic logic. *Mind*, 60:1–15.

[Walker and Wooldridge, 1995] Walker, A. and Wooldridge, M. (1995). Understanding the emergence of conventions in multi-agent systems. In *Proceedings of the First International Conference on Multi–Agent Systems*, pages 384–389, San Francisco, CA. MIT Press.

[Watts and Strogatz, 1998] Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of small-world networks. *Nature*, 393(6684):440–442.

[Wooldridge, 2002] Wooldridge, M. (2002). *Introduction to MultiAgent Systems*. John Wiley & Sons.

[Xiong et al., 2004] Xiong, L., Liu, L., and Society, I. C. (2004). Peertrust: supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Transactions on Knowledge and Data Engineering*, 16:843–857.

[Yarbrough and Yarbrough, 1999] Yarbrough, B. V. and Yarbrough, R. M. (1999). Governance structures, insider status, and boundary maintenance. *Journal of Bioeconomics*, 1:289–310.

[Yolum and Singh, 2002] Yolum, P. and Singh, M. (2002). Flexible protocol specification and execution: Applying event calculus planning using commitments. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 527–534. ACM Press.

[Younger, 2004] Younger, S. (2004). Reciprocity, normative reputation, and the development of mutual obligation in gift-giving societies. *Journal of Artificial Societies and Social Simulation*, 7(1).

[Younger, 2005] Younger, S. (2005). Reciprocity, sanctions, and the development of mutual obligation in egalitarian societies. *Journal of Artificial Societies and Social Simulation*, 8(2).

[Yu and Singh, 2000] Yu, B. and Singh, M. P. (2000). A social mechanism of reputation management in electronic communities. In *CIA '00: Proceedings of the 4th International Workshop on Cooperative Information Agents IV, The Future of Information Agents in Cyberspace*, pages 154–165, London, UK. Springer-Verlag.

[Yu et al., 2008] Yu, H., Gibbons, P. B., Kaminsky, M., and Xiao, F. (2008). Sybillimit: A near-optimal social network defense against sybil attacks. In *SP '08: Proceedings of the 2008 IEEE Symposium on Security and Privacy*, pages 3–17, Washington, DC, USA. IEEE Computer Society.

[Yu et al., 2006] Yu, H., Kaminsky, M., Gibbons, P. B., and Flaxman, A. (2006). Sybilguard: defending against sybil attacks via social networks. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 267–278, New York, NY, USA. ACM.