A Peer-to-Peer Normative System to Achieve Social Order

Amandine Grizard, Laurent Vercouter, Tiberiu Stratulat, Guillaume Muller

▶ **To cite this version:**

Amandine Grizard, Laurent Vercouter, Tiberiu Stratulat, Guillaume Muller. A Peer-to-Peer Normative System to Achieve Social Order. COIN: Coordination, Organisation, Institutions and Norms in agent systems, Aug 2006, Riva del Garda, Italy. pp.274-289, 10.1007/978-3-540-74459-7_18 . lirmm-00102798

HAL Id: lirmm-00102798

https://hal-lirmm.ccsd.cnrs.fr/lirmm-00102798

Submitted on 2 Oct 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A peer-to-peer normative system to achieve social order

Amandine Grizard[1], Laurent Vercouter[2], Tiberiu Stratulat[3], and Guillaume Muller[2]

[1] Institut Eurecom, Affective Social Computing Lab.,
2229 routes des crêtes, BP 193, F-06904 Sophia Antipolis, France
grizard@eurecom.fr
[2] École N.S. des Mines de Saint-Étienne, Multi-Agent System Dpt
158 cours Fauriel, F-42023 Saint-Étienne Cedex 02, France
{vercouter,muller}@emse.fr
[3] LIRMM
161 rue Ada, F-34392 Montpellier Cedex 5, France
stratulat@lirmm.fr

Abstract. Social order in distributed descentralised systems is claimed to be obtained by using social norms and social control. This paper presents a normative P2P architecture to obtain social order in multi-agent systems. We propose the use of two types of norms that coexist: rules and conventions. Rules describe the global normative constraints on autonomous agents, whilst conventions are local norms. Social control is obtained by providing a non-intrusive control infrastructure that helps the agents build reputation values based on their respect of norms. Some experiments are presented that show how communities are dynamically formed and how bad agents are socially excluded.

Introduction

In multi-agent systems the execution of global tasks strongly differs according to the centralised or decentralised nature of the system. Decentralisation implies that information, resources and agent capacities are distributed among the agents of the system and hence an agent cannot perform alone a global task. The most popular examples of decentralised multi-agent systems are peer-to-peer (P2P) networks used for file sharing. In such applications, agents must collaborate to index shared files and propagate queries for given files. It is also essential that all agents use compatible strategies for propagation (in most of the cases they use the same strategy) to ensure the correct termination of search algorithms.

Peer-to-peer systems illustrate how important it is in decentralised systems that each agent behaves well, that is to be compliant to some expected "good" behaviour, in order to cooperate with the others. If not, the activities of other agents can be blocked or corrupted. In peer-to-peer systems we usually consider that the agents have been downloaded from the same place and are coded by

the same developers. It is then natural to consider that the agents will behave the same as expected. But if we consider decentralised multi-agent systems in general, this assumption is not realistic. Heterogeneity and autonomy are the required properties of the agents to build open and flexible systems and they rule out any assumption concerning the way they are constructed and behave. Moreover, the agents can no longer be controlled by central institutions which supervise their behavior since we consider decentralised systems.

C. Castelfranchi [1] also claimed that, in decentralised system, Social Order is achieved by the use of norms as rules of good behaviour and through *Social Control*. In order to preserve the openness and the flexibility of the system, norms are only external representations that should not be hard coded into the agents, since they are supposed to be dynamically created or modified. Norms also preserve the autonomy of the agents, since smart agents can reason on them, decide autonomously to respect or violate them and observe the behaviour of other agents if they are norm compliant. Social Control mainly refers to the fact that each agent is observed and controled by some other agents from the same system. However in the literature we find mainly trust mechanisms that are proposed to achieve it [2–4]. According to these works agents with bad behaviors are punished by social sanctions, get bad reputations and are excluded from the society.

In this paper we propose a peer to peer normative architecture to obtain social order in a decentralised system. The main contribution of our proposition is not on the formalism used to represent the norms nor on the representation and calculation of reputation but rather on the *integration* of norms within a trust model that allows agents to perform Social Control. We also provide some mechanisms to update existing norms when groups of agents feel the need for the system to evolve. The next section defines the concepts of norms used in this article and describe their formalization. Section 2 describes the architecture of a decentralised multi-agent system that performs social control and some experimental results are presented in section 3. Finally, we propose in section 4 a mechanism to adapt the content of some specific types of norms, called conventions and we conclude in the last section.

1 Norms and control

In the area of agent-based systems, two important contradictory properties are needed: autonomy and control. Autonomy abstracts out the way an agent behaves when asked to solve a problem or execute a task. Control is necessary to be sure that autonomous agents behave according to the specifications formulated by someone on their behavior. In terms of degrees of freedom, the former relieves, the latter constrains.

Norms have been recently considered as being good candidate tools to design agent-based system and also to solve the paradoxical problem that confronts the preservation of the autonomy of the agents and the need of control. Getting its inspiration from social sciences, a norm is mainly a description of an (ideal)

behavior that an agent or a group of agents is expected to display. The normative behavior is generally described by using deontic constraints, such as obligations, permissions and interdictions. Although there is no complete agreement on how to use norms in artificial systems, there are however two main trends [5]. The first considers a norm as a sort of specification of good behavior that, once identified, should be hard coded directly into the agents. The agents are by construction norm compliant or "regimented". The second trend adopts a more flexible perspective, where norms are considered as being only indications of an ideal behavior that could be adopted or not by an agent. In the case where the norm is not respected we talk about violation. In order to avoid the proliferation of unexpected behavior and therefore the chaos in the system, some of the works adopting the second perspective suggest the use of various structures that allow to sanction the deviating behavior and hence to "control" the agents.

In the literature two main categories of control structures are described: internal and external. Internal control is where an agent is able to identify by itself which is the good behavior to adopt according to an external reference. In this type of scenario we count on the agents' cognitive and learning capacities to understand and evaluate the normative behavior.

External control is where some external or institutional structures can interfere (even physically) with the agent and hence influence it to display the normative behavior. The behavior of an agent is interpreted and evaluated by others (authorities, group of agents, etc.) according to the norms governing the system wherein the agent acts.

1.1 Social Norms revisited

In the domain of social sciences, R. Tuomela proposed a theory of social norms that characterizes communities in human societies [6]. For him a social norm has the form "An agent of the kind F in group G ought to perform task T in situation C". Such norms are further divided into rules (r-norms) and proper social norms (s-norms). The r-norms are the norms created by an authority or a body of agents that are authorized to represent the group. The obedience to the rules is made on an agreement-making basis and their violation is explicitly sanctioned. The governmental laws are examples of r-norms. The s-norms represent the conventions or the mutual beliefs about the right thing to do in a community. An agent obeys also to an s-norm because it believes that the other members of the community expect that. The sanction of an s-norm is only social: approval or disapproval and it can not be decided in advance.

The definition of both types of norms is given in terms of certain conditions. For instance a norm N of obligation which says "Everyone in G ought to perform task T when in situation C" is considered an r-norm iff the following conditions are satisfied (see [6] for a detailed discussion):

- *promulgation* condition: N has been issued by an authority;
- *accessibility* condition: the agents are aware that N is in force;
- *pervasiveness* condition: many members of G perform T in C;

– *motivational* condition: some of the agents that perform T do that because they believe that they ought to perform T in C;
– *sanction* condition: there are sanctions applied when N is violated.

In the case of an s-norm, the following conditions should be satisfied:

– *acceptance* condition: there is a mutual belief in G to the effect that the members of G ought to perform T in situation C;
– *pervasiveness* condition: many members of G perform T in C;
– *motivational* condition: some of the agents that perform T do that because they believe that they ought to perform T in C and that is what is expected by other members of G;
– *recognized sanction* condition: there are social sanctions applied when N is violated.

There are two main cases where s-norms and r-norms coexist: parallel norms (norms with the same content) and conflicting norms (norms with conflicting content). Tuomela considers that in both cases r-norms override s-norms but which one or ones of these kinds of social norms wins empirically is a factual issue not to be decided a priori.

Tuomela shows further that s-norms characterize the high-trust societies whereas r-norms are norms for low-trust, or business-like societies.

Since Tuomela's theory of social norms stands for human societies we will try to adapt it to be applied to artificial societies of agents. Therefore we propose in this article to use two different but coexisting settings that correspond to the use of r-norms and s-norms.

The first setting is related to the use of r-norms. We will reconsider the definition of an r-norm, that we will call simply a *rule* in the rest of the article. A rule R of obligation is described by using the following notation:

$$rule(C, T, S_+, S_-)$$

which says that in the context described by C, T ought to be executed. If T is the case the agent is rewarded with S_+ otherwise it gets sanctioned with S_-.

In addition to the adoption of such a notation for rules, the setting we propose should satisfy the other conditions given in the definition of an r-norm. The architecture we describe in the following section contains as main ingredients:

1. authorities responsible to create independent sound sets of norms (promulgation condition) and to inform the autonomous agents about them (accessibility condition);
2. ordinary autonomous agents that receive the norms and that are observed and controlled if they are norm compliant;
3. monitoring or control structures that evaluate the behavior of the agents and sanction or reward them according to the normative content (sanction condition).

The pervasiveness condition in the definition of an r-norm accounts for the supposed fact that people in human societies tend to obey the norms. In the setting we propose, the pervasiveness condition is relaxed and is sometimes obtained by applying the sanctions. But it is not guaranteed because of the hypothesis we made to respect the autonomy of the agents. The motivational condition is also very strong, because it considers that i) the agents have cognitive abilities and that ii) they adopt the normative behavior rationally. The hypothesis we adopt on the agents we use is hence that they are autonomous decision-makers able to decide by themselves what to do next, given their internal state, the normative information and the current state of their environment.

The second setting is about s-norms. In a similar way, we propose to re-adapt the definition of an s-norm that we will simply call *convention*. The notation we use for the conventions of obligation is as follows:

$$convention(C, T)$$

This notation says that in the context where C is true an agent ought to do the task T.

As in the case of rules, the setting we propose should allow a community of agents to create and manipulate conventions. The conditions that define an s-norm are therefore partially preserved. For instance, the acceptance condition is obtained by introducing a protocol that allows some agents to exchange information about the current mutual convention. Concerning the recognized sanction condition, since this setting coexists with the setting for rules, we will use the same monitoring structures to help an agent observe the behavior of the other agents. However, the decision on how to sanction or reward the others is made only by the concerned agent, and can not be known in advance. For the pervasiveness and motivational conditions, we make the same hypothesis that we made for rules.

1.2 Reputation as sanction or reward

One of the differences existing between rules and conventions consists in the application of sanctions. As shown above, two explicit sanctions are defined in the specification of a rule: a positive sanction applied when an agent respects the deontic constraint and a negative sanction applied in the contrary case. In most of normative systems, these sanctions are material, for instance a fee. In our case, we do not use material sanctions because it implies that there is a way to enforce an agent to pay a fee or at least a physical means of pressure on the agent. This assumption is contradictory with the property of autonomy claimed for agents and it is even more the case for decentralised systems because agents may be deployed on different platforms.

We propose to describe sanctions as a positive or negative influence on the reputation of an agent. Agent reputation is a concept that has been studied in several works [7, 2, 3, 8, 4] with different approaches but also with different semantics attached to reputation. The word "reputation" does not have exactly

the same meaning depending on the considered approach. Some authors consider that an agent has only one reputation maintained globally by the system [9], whereas others think that two agents can have a different opinion about the reputation of an agent [2, 8, 4]. Moreover, some works consider that reputation should be relative to a given context [7], to the sources used to build it [10], to the nature of its target [10], to the estimated facet of the agent [2, 11], ... A unified view of all these aspects has recently been proposed in a functional ontology of reputation [12]. However, we consider reputation more generally in this paper. Since our contribution is not on the reputation model but on the integration of the concepts of reputation and of norms in a peer-to-peer environment, we only represent reputation by a simple plain value. We will consider using more precise concepts of reputation in future works. The only property that is important here is that two agents can maintain different reputation values for a same target as there is no global view or control of the system.

Thus, reputations are values maintained by other agents and are external to the sanctionned agent. The violation of a rule will cause a decrease of the agent's reputation whereas positive sanction may cause its increase. Sanctions in term of reputation are still an incentive to respect the rules for the agents because one of the consequences of a low reputation for an agent could be the refuse of certain other agents to interact with it (social exclusion). Such sanctions can be viewed as social constraints on the agents.

We consider that the sanction for a rule is defined in terms of reputation and we formalize it as follows:

$$sanction(Applier, Sanctioned, Weight)$$

where *Applier* is the agent applying the sanction, *Sanctioned* is the agent sanctioned and *Weight* represents generally the value of the sanction. We use the last parameter to affect the reputation of an agent according to a mechanism which is explained in latter sections.

In the case of conventions, the sanction is not represented explicitly because it can not be known a priori. It depends on the agent in cause and it is the expression of an approval or disapproval of the actual behavior. It is up to an agent to sanction or not other agents that violate or respect its own conventions. These sanctions may also impact reputation values but these values should be maintained locally by the owner of the convention. In this case the scope of the sanctions does not cover the whole society but only the relation between one agent and another agent or group of agents.

2 An overlay system to sanction the violation of rules

Since agents are autonomous, we can not assume that they will respect the rules of their system. There must be a way to observe their behavior, their compliance to the rules and to sanction the violations. If such a mechanism exists and is efficient, rational agents are more motivated to respect the rules in order to avoid being sanctionned. This is achieved by a social control of the system if it is the

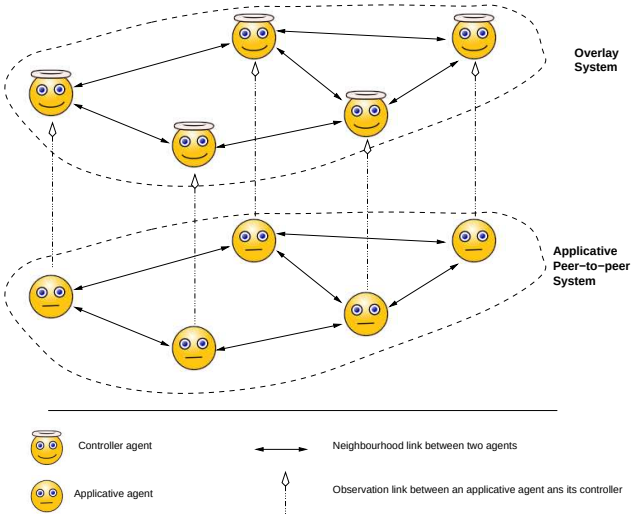

Fig. 1. Overview of the overlay system

society of agents that supervises and sanctions its members. Then, there is a need for a control system that can be integrated to an existing multi-agent system. This integration can only be effective if the control system takes into account the specific features of a multi-agent system, in particular the decentralised nature of the system and the autonomy of the agents.

In this section, we propose and describe such a control system for a P2P system of *application agents* in order to control the respect of the rules governing the system. To comply to the features quoted above, the control system is also a multi-agent system where each *controller agent* is associated to an application peer of the controlled P2P system (see Figure 1). The control system is an overlay system in which each controller agent has a partial view of the interactions inside the P2P system. Based on observations on the application agents, a controler agent detects rule violations and cooperates with other controller agents in order to sanction the violators in the P2P system. This solution keeps the decentralised nature of the whole system and is not intrusive for the peers. By "non-intrustive", we mean that there is no constraint on the agent's internal implementation. The observation of the agent's communication is not an intrusion since communications are sent and transit through an interaction medium and we assume that this medium can be observed. This assumption remains realistic because one of the property of communications is that they are public as claimed in [13].

Reputation is used as a mean to sanction the application agents. Thus the reputation of an application agent that violated a rule will decrease for each other controller agent that has observed and detected the violation. Then, the controller agent can share its reputation models with other controller agents or

even with its application agent to which it is connected. The reputation of the application agents is hence locally propagated and application agents that do not respect the rules can get bad reputations and be excluded by other application agents from future interactions. We consider that the overlay system performs a social control over the application P2P system and that it aims at excluding the application agents that violate the rules.

2.1 Application agents

The tasks performed by application agents are completely dependent of the kind of P2P system considered. For instance, in the case of P2P networks used for file sharing, application agents possess some files to share and are able to formulate queries for given files or answer to or propagate the queries from other agents. The behavior of application agents is application-dependent and, since they are autonomous, we do not make any assumption on it. However, inside the system, an application agent may behave or not as it is expected by other application agents.

The goal of the overlay system is to control that application agents respect the rules of the system. The rules only focus on the communicative behavior of application agents. Application agents should be considered as black boxes and there is no way to observe their internal functioning to check if they comply to the rules. The only thing we can control are the external actions of an application agent, which correspond to the interactions with other agents. We therefore make the assumptions that (i) interactions between application agents can be observed by some controller agents; (ii) there exists some rules, formalised as described in section 1, that application agents must respect; (iii) these rules are available for both application and controller agents.

The neighborhood of an application agent is composed of other application agents. We do not study the reason why some application agents are neighbors or not (it also depends on the application) and we note $Neighbors(App_i)$ the set of neighbors of the application agent App_i.

2.2 Controller agents

The overlay system is composed of controller agents. A controller agent has the capacity to observe the interactions of an application agent, that is the messages sent and received by this agent. It also knows the rules that must be respected by application agents and can detect violations by comparing its observations to the rules. At last, a controller agent has the capacity to sanction an application agent, and to do so, it may have to collaborate with other controller agents. We should also note that we assumed that the controller agents are deployed by a trusted third party and that they are trustworthy.

Each application agent is associated to a unique controller agent in the overlay system described here. This association preserves the autonomy of the application agent and is not very constraining for the P2P system. For example, we can imagine in a P2P file sharing system that controller agents are hosted

by internet providers and that they can observe the messages exchanged. From the point of view of an application agent, the existence of a controller is a real advantage since it provides the information about the reputation of the other application agents in its neighborhood. Then the application agent can use this information to choose to cooperate with application agents with good reputation.

The system we propose is caracterized by the fact that for an application agent App_i the neighbors of the associated controller $Cont_i$ is the set of controllers associated to the neighbors of agent App_i:

$$App_j \in Neighbors(App_i) \equiv Cont_j \in Neighbors(Cont_i)$$

The reputation model used by a controller agent considers the reputation value for each application agent in the neighborhood of its application agent. We note Rep_i^j the reputation of an agent $App_j \in Neighbors(App_i)$ computed by the controller $Cont_i$.

2.3 Interaction between agents

between application agents An application agent interacts with its neighbors (other application agents). The nature and the content of these interactions depends on the application and we do not make any assumption over them. However, application agents should respect the rules of the system if they do not want to be sanctionned.

between application and controller agents An application agent interacts with its own controller agent and can not interact with other controller agents. Such interactions are used by an application agent to get some information related to norms or reputation. Application agents may ask two types of information to their controller agent: (i) the rules that they must respect in their interaction with other application agents; (ii) the reputation value of one or several of its neighbors. These interactions are only possible and not required. Application agents are autonomous and we can not assume that they will interact in a given way with the controller. This possibility of interaction enables an application agent that could reason on norms and reputation to get the information necessary for its reasoning. However, it is also possible to consider different application agents that do not interact with their controller but, if the respect of norms is not hardcoded in their implementation, they are likely to be sanctionned. A controller agent does not need to interact with its application agent but it is able to oversee its interactions and check for violations.

between controller agents A controller agent interacts with other controller agents to inform them when its application agent violated a rule or behaved well. For instance, when a controller agent detects that its application agent has violated a rule, it sends a message to its neighbors that are concerned by the violation in order to ask them to sanction its application agent using the

negative sanction field of the rule. At the opposite, it can also ask its neighbors to apply the positive sanction if the application agent has respected the rule.

3 Experimental results

The normative system presented here has been implemented and tested on a scenario of a P2P file sharing network.

3.1 The scenario of P2P file sharing

We consider a P2P system where each application agent possesses some files to share with other agents and needs to download some files that may be contained by other peers. For simplicity, we used a gnutella-like [14] protocol. According to this protocol, an agent that requests a file formulates a query with the file name and sends this query to each of its neighbors. Each neighbor looks if it owns a file that matches the query. If it does, it responds positively to the agent requester. If not, it does not respond. Then, each neighbor propagates further the query to its own neighbors.

To avoid flooding the network, the queries are enriched with two fields. First, the queries have an unique ID so that an agent does not have to handle twice the same query. The second field is called TTL (*Time To Live*). The TTL corresponds to the depth of propagation of the query in the network. For example a query with a TTL of 3 will be propagated to the neighbors of the neighbors of the neighbors of the requester but not further. When an agent receives a query, it decreases its TTL and it propagates it to its neighbors only if the value of the TTL is at least 1.

3.2 Rules of the scenario

The correct functionning of a P2P file sharing system requires that the agents propagate the queries of other agents. This is a cooperative behavior, but selfish agents may not behave like this because it would cost them some ressources (CPU time) consumed for the benefit of others. We propose to define the following rule as an incentive to behave cooperatively:

$$
\begin{aligned}
&rule(asked(Id, TTL, App_b, App_s, App_r, file) \wedge TTL > 1 \wedge trusted(App_s),\\
&\quad asked_neighbors(Id, TTL - 1, App_r),\\
&\quad \emptyset,\\
&\quad sanction(Cont_r, App_r, HigherRep(Cont_s, S_+), LowerRep(Cont_s, S_-)))
\end{aligned}
$$

This rule is an obligation in the context $asked(Id, TTL, App_b, App_s, App_r, file)$ representing that a query about the file $file$ initiated by the agent App_b with the id Id has been received by the agent App_r from the agent App_s with a TTL greater than 1. This context also requires that the sender of the query App_s is trustworthy. This condition is necessary to avoid that an agent that do not want

to interact with untrusted neighbors is sanctionned by its controller. The trusted or untrusted nature of a neighbor can be deduced by the controller according to its reputation about App_s, and it may be communicated to App_r if it requests it.

In this context, a task must be achieved to obtain $asked_neighbors(Id, TTL-1, App_r, file)$ that means that the query Id should be propagated to the neighbours of App_r with a TTL decreased by 1. If this obligation is respected the positive sanction indicates that the controller $Cont_r$ will ask to the controller $Cont_s$ to increase the reputation of App_r by a value S_+. Otherwise the negative sanction $sanction(Cont_r, App_r, LowerRep(Cont_s, S_-))$ is applied. This negative sanction indicates that the controller $Cont_r$ will sanction its application agent App_r by asking to the controller $Cont_r$ to lower the reputation of App_r by a value S_-.

3.3 Experiments

Some experiments have been done to observe how the reputation of an agent evolves if it does not respect the rules. We used PeerSim simulator [15] to simulate the P2P protocols, Java to implement the agents and Prolog with JPL [16] to code and interpret the rules.

The tests have been done on a set of 50 agents. Each agent owns from 5 to 10 files taken from a global set of 100 different files. The agents are connected in a network such that each agent has a minimum of 2 and a maximum of 5 neighbors. The simulation lasts 15 cycles. At each cycle an agent formulates from 3 to 5 queries for a file (randomly chosen) and sends on query to its neighbors with a TTL of 2. All queries are propagated according to the gnutella protocol and the agents that own a file matching a received query, answer positively to the requester.

The controller of an agent computes the reputation of the neighbors of its agent by using values in the domain [0:1] with an initial value of 0.8. The positive and negative sanctions associated to the rule are $S_+ = 0.01$ and $S_- = -0.02$. Initially, all the agents behave well and respect the rule. After 5 cycles, one agent (the agent 0 in the figures) changes its behavior: it will not respect systematically the rule and in 50% of the cases, it does not propagate the queries from its neighbors. This violation of the rule is detected by its controller which applies the negative sanction. The impact of the sanction is shown in figure 2 representing the average value of the reputation value of agent 0 kept by the controllers of its neighbors.

The reputation value of an agent is used by an application agent to remove that agent from its neighborhood, for instance, when it has a low reputation. Removal of a neighbor means that queries from it will no longer be considered (neither answered, nor propagated) and that queries from other agents will not be sent further to the neighbor with a bad reputation. This is equivalent to an irrevocable social exclusion, since the removed neighbor will not have the possibility to get positive sanctions and then to recover an acceptable reputation.

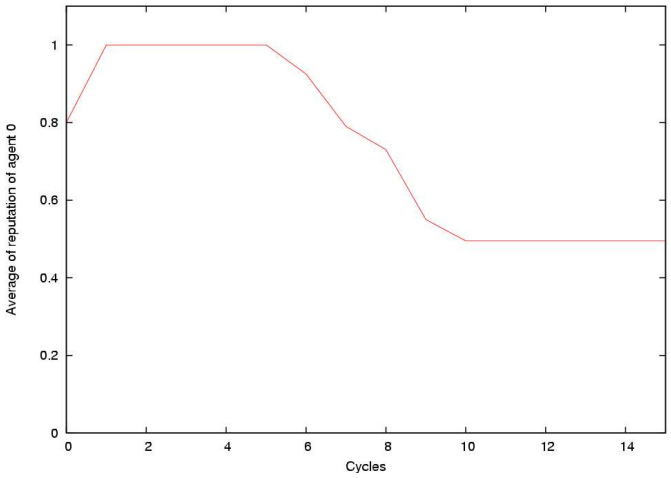

Fig. 2. Average of the reputation of an agent (violations occur at step 5)

If we wanted to keep the possibility to "forgive" to an agent, we may have kept the possibility for it to get positive sanctions.

The exclusion of agent 0 can be seen on figures 3 and 4. Figure 3 represents the number of queries sent by agent 0 that are propagated by the rest of the network. We can see that this number decreases when other agents begin to remove agent 0 from their neighborhood (around cycle 9) and that no more queries from it are considered at cycle 12. Figure 4 shows the percentage of files received by agent 0 at each cycle. Since its queries are no more considered, this percentage begins to decrease at cycle 9 and reaches 0 at cycle 11.

The reputation threshold, below which an agent removes a neighbor from its neighborhood, has been set to 0.5 for the simulations. This explains the fact that the reputation value of agent 0 does not continue to decrease in figure 2 after its exclusion, since the agents stop interacting with it and therefore agent 0 can not violate the rule anymore.

4 Convention dynamics

The control system presented in the section 2 can be used to enforce the respect of the rules of the system. If an agent does not respect these rules, its reputation become lower and lower and other agents will stop cooperating with it. In section 1, we mentioned the distinction we make between two kinds of norms: rules and conventions. The main difference is that a rule is shared by the whole system and its violation should be explicitly sanctionned whereas conventions rather refer to an usage local to a group of agents. Conventions describe the behavior

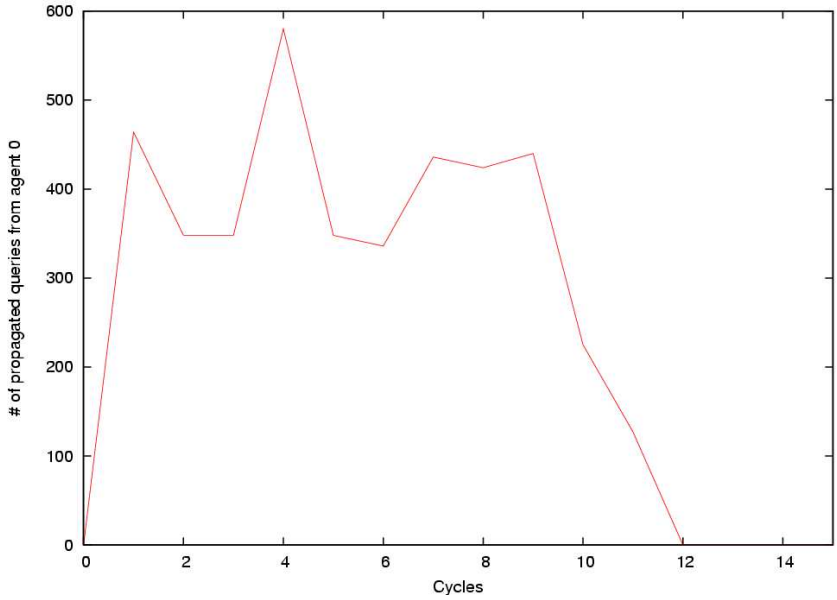

Fig. 3. Number of propagated queries from the agent violator

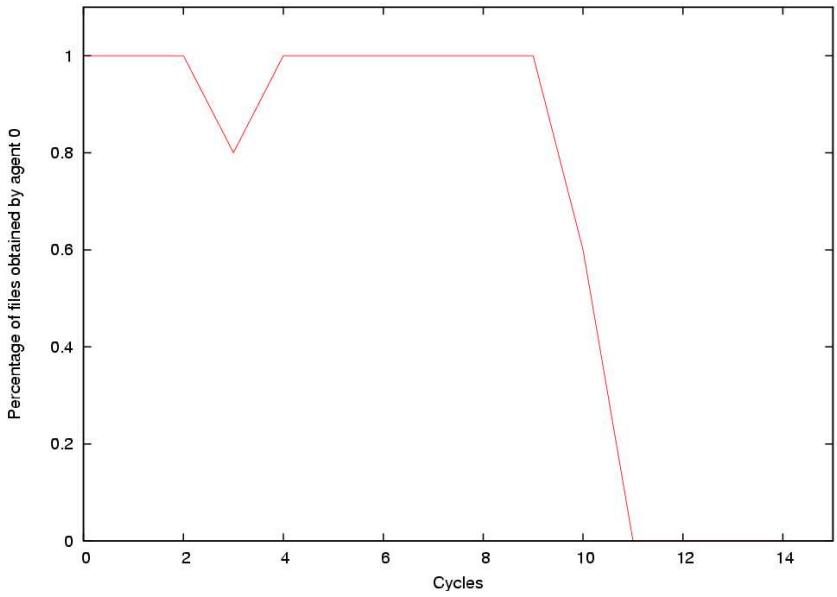

Fig. 4. Percentage of files got by the agent violator

approved and expected by the group. In case of violation of the convention, the sanction is not explicit.

In the example of file sharing P2P networks, described in section 3, a convention can establish, for instance, the initial value of the TTL of a query. The higher is the TTL, the better are the chances to get the required file. But higher values of TTL also means that there will be more communications needed to propagate the query and that the network will be more loaded and that it will be slowed. We can continue to imagine that some agents request rare files difficult to obtain and therefore require a high TTL value, while agents that send many queries for commonly shared files would prefer a low TTL value.

Therefore, we can have many cases showing different conventions needed only by some agents that could regroup in small communities according to their preferences. Our proposal is therefore to use conventions to reorganize the neighborhood links between agents in order to group agents sharing the same conventions. We propose to use the overlay system presented in section 2 to deal with conventions as it follows:

- an application agent App_i sends its own set of conventions $Conventions(App_i)$ to its controller agent $Cont_i$.
- the controller $Cont_i$ informs its neighbors (belonging to the set $Neighbors(Cont_i)$) that its application agent has the set of conventions $Conventions(App_i)$.
- each controller $Cont_j \in Neighbors(Cont_i)$ keeps this information available for its application agent App_j.

The controller $Cont_i$ now observes the behavior of other agents to check if they violate or not a convention of App_i. If a violation occurs, $Cont_i$ does not apply any sanction but inform App_i of this violation. The sanction to apply is up to App_i. We suggest that it also maintains a reputation model about its neighbors. This reputation is attached to conventions and should not replace the reputation based on the respect of rules. Then, two reputation values co-exist for each neighbor of App_i. A violation of a convention may result in a decrease of the corresponding reputation value.

The interpretation by App_i of its reputation model attached to conventions is free, but here again we suggest that it uses it to modify its neighborhood. For instance, App_i could remove agents with a low reputation value for conventions, from its neighborhood and then look for other agents to replace them in its neighborhood. The method used to find new neighbors depends on the kind of P2P network considered and is anyway out of the scope of this paper. The modification of the agents' neighborhood brings a reorganisation of the P2P system and communities of agents sharing similar conventions should be constituted. This reorganisation can also be dynamical because an agent can change its conventions and then progressivly change its neighborhood.

This is ongoing work and an implementation of such a control system that includes conventions is in progress.

5 Conclusion

The work described in this article follows the guideliness proposed by Castelfranchi in [1] who claims that social order in distributed decentralised systems is to be obtained by using social norms and social control. Concerning the use of norms the adaptation of the concepts of social norms proposed by Tuomela (for human societies) to multi-agent systems seems to satisfy the need of both types of control, centralised and decentralised. We introduced the concept of rules to describe the global norms that constrain globally all the agents of a system, and conventions to describe the rules that could be created and applied locally to a group of agents by mutual acceptance.

Concerning the manifestation of social control, we showed how to obtain it in a P2P network of application agents. Each application agent decides autonomously with whom to cooperate based on the reputation values of other agents. The reputation value depends on the respect or not of the rules and conventions that are currently in force. It is the role of the overlay system to monitor the behaviours and inform the application agents about the reputation of the others. The control of the agents is completely non-intrusive and decentralised.

The experiments showed interesting results, notably the fact that agents with good reputations are rapidly identified and are at the center of communities of agents with similar behaviour. In future works we would like to study the dynamics of open systems when agents come and go and when the content of the norm changes dynamically.

References

1. Castelfranchi, C.: Formalising the informal? dynamic social order, bottom-up social control, and spontaneous normative relations. Journal of Applied Logic **1** (2003) 47 – 92
2. Sabater, J., Sierra, C.: REGRET: reputation in gregarious societies. In Müller, J.P., Andre, E., Sen, S., Frasson, C., eds.: Proceedings of the Fifth International Conference on Autonomous Agents, Montreal, Canada, ACM Press (2001) 194–195
3. Conte, R., Paolucci, M.: Reputation in Artificial Societies. Social Beliefs for Social Order. Volume 6. Springer (2002)
4. Muller, G., Vercouter, L.: Decentralized monitoring of agent communication with a reputation model. Trusting Agents for trusting Electronic Societies, Lecture Notes in Computer Science (3577) (2005) 144–161
5. Jones, A.J.I., Sergot, M.: On the characterisation of law and computer systems: The normative systems perspective. In Meyer, J.J.C., Wieringa, R.J., eds.: Deontic Logic in Computer Science: Normative System Specification. John Wiley & Sons (1993)
6. Tuomela, R.: The Importance of Us: A Philosophical Study of Basic Social Norms. Stanford University Press (1995)
7. Castelfranchi, C., Falcone, R.: Principles of trust in mas: cognitive anatomy, social importance and quantification. In: ICMAS'98, Paris (1998) 72–79
8. Abdulrahman, A.: A framework for decentralized trust reasoning. PhD thesis, University of London (2004)

9. Zacharia, G., Moukas, A., Maes, P.: Collaborative reputation mechanisms in electronic market places. In: Proceedings of the 32th Hawaii International Conference on System Sciences. (1999)
10. McKnight, D., Chervany, N.: Trust and distrust definitions: One bite at a time. In: Proceedings of the AAMAS'01 Trust in Cyber-societies workshop, Springler-Verlag Berlin Heidelberg (2001) 27–54
11. Wang, Y., Vassileva, J.: Bayesian network-based trust model in peer-to-peer netwoeks. In: Proceedings of the Workshop on Deception, Fraud and Trust in Agent Societies. (2003) 57–68
12. Casare, S., Sichman, J.: Using a functional ontology of reputation to interoperate different agent reputation models. Journal of the Brazilian Computer Society **11**(2) (2005) 79–94
13. Singh, M.P.: Agent communication languages: Rethinking the principles. In Huget, M.P., ed.: Communication in Multiagent Systems. Volume 2650 of Lecture Notes in Computer Science., Springer (2003) 37–50
14. Gnutella: Gnutella 0.48 specifications rfc (2000) http://rfc-gnutella.sourceforge.net/developer/stable/.
15. PeerSim: A peer-to-peer simulator (2005) http://peersim.sourceforge.net/.
16. JPL: A java interface to prolog (2003) http://www.swi-prolog.org/packages/jpl/java_api/.