Users' Collaboration as a Driver for Reputation System Effectiveness: a Simulation Study

Guido Boella and Marco Remondino
Department of Computer Science, University of Turin
boella@di.unito.it , remond@di.unito.it

Gianluca Tornese (for the implementation)
gianluca.tornese@libero.it

Abstract

Reputation management is about evaluating an agent's actions and other agents' opinions about those actions, reporting on those actions and opinions, and reacting to that report thus creating a feedback loop. This social mechanism has been successfully used, through Reputation Management Systems (RMSs) to classify agents within normative systems. Most RMSs rely on the feedbacks given by the member of the social network in which the RMS itself operates. In this way, the reputation index can be seen as an endogenous and self produced indicator, created by the users for the users' benefit. This implies that users' participation and collaboration is a key factor for the effectiveness a RMS. In this work the above factor is explored by means of an agent based simulation, and is tested on a P2P network for file sharing.

1. Introduction

In everyday's life, when a choice subject to limited resources (like for instance money, time, and so on) must be done, due to the overwhelming number of possibilities that people have to choose from, something is needed to help them make choices. People often follow the advice of others when it comes to which products to by, which movies to watch, which music to listen, which websites to visit, and so on. This is a social attitude that uses others' experience They base their judgments of whether or not to follow this advice partially upon the other person's reputation in helping to find reliable and useful information, even with all the noise.

Using and building upon early collaboration filtering techniques, reputation management software gather ratings for people, companies, and information sources. Since this is a distributed way of computing reputation, it is implicitly founded on two main assumptions:

1) The correctness of shared information
2) The participation of users to the system

While the negation of the first could be considered as an attack to the system itself, performed by users trying to crash it, and its occurrence is quite rare, the second factor is often underestimated, when designing a collaborative RMS. Users without a vision of the macro level often use the system, but simply forget to collaborate, since this seems to cause a waste of time.

The purpose of the present work is to give a qualitative and, when possible, quantitative evaluation of the collaborative factor in RMSs, by means of an empirical analysis conducted via an agent based simulation. Thus, the main research question is: what's the effectiveness of a RMS, when changing the collaboration rate coming from the involved users?

In order to answer this question, in the paper an agent based model is introduced, representing a peer-to-peer (P2P) network for file sharing. A basic RMS is applied to the system, in order to help users to choose the best peers to download from. In fact, some of the peers are malicious, and they try to exploit the way in which the P2P system rewards users for sharing files, by uploading inauthentic resources when they do not own the real ones. The model is described in detail and the results are evaluated through a multi-run *coeteris paribus* technique, in which only one setting is changed at a time. In particular, the most important parameters which will be compared, to evaluate the effectiveness of the RMS are: verification of the files, performed by the users and negative payoff, given in case a resource is reported as being inauthentic. The verification of the files, i.e. users' the collaboration, is an exogenous factor for the RMS, while the negative payoff is an endogenous and thus directly controllable factor, from the point of view of a RMS's designer.

The P2P framework has been chosen since there are many works focusing on the reputation as a system to overcome the issue of inauthentic files, but, when evaluating the effectiveness of the system, the authors [1] usually refer to idealized situations, in which users always verify the files for authenticity, as soon as they start a download. This is obviously not the case in the real world: first of all, most resources require to be at least partially owned, in order to be checked. Besides, some users could simply decide not to check them for long time. Even worse, other users could simply forget about a downloaded resource and never check it. Last but not least, other users might verify it, but simply not report anything, if it's not authentic.

2. Reputation and P2P Systems

Since uploading bandwidth is a limited resource and the download priority queues are based on a uploading-credit system to reward the most collaborative peers on the network, some malicious users create inauthentic files, just to have something to share, thus obtaining credits, without being penalized for their behavior. To balance this, RMSs have been introduced, which dynamically assign to the users a reputation value, considered in the decision to download files from them or not. RMSs are proven, via simulation, to make P2P networks safe from attacks by malicious peers, even when forming coalitions. In networks of millions of peers attacks are less frequent, but users still have a benefit from sharing inauthentic files. It's not clear if RMSs can be effective against this selfish widespread misbehavior, since they make several ideal assumptions about the behavior of peers who have to verify files to discover inauthentic ones. This operation is assumed to be automatic and with no costs. Moreover, since the files are usually shared before downloading is completed, peers downloading inauthentic files unwillingly spread them if they are not cooperative enough to verify their download as soon as possible. In the present work, the creation and spreading of inauthentic files is not considered as an attack, but as a way in which some agents try to raise their credits, while not possessing the real resource that's being searched by others. A basic RMSs is introduced, acting as a positive or negative reward for the users and human factor behind the RMSs is considered, in the form of costs and benefits of verifying files. Most approaches, most notably EigenTrust [2], assume that verification is made automatically upon the start of download of the file. By looking as we do at the collaboration factor in dealing with RMSs, we can question their real applicability, an issue which remains unanswered in the simulation based tests made by the authors. To provide an answer to this question it is necessary to build a simulation tool which aims at a more accurate modeling of the users' behavior rather than at modeling the reputation system in detail.

3. Model Framework

We assume a simple idealized model of reputation, since the objective is not to prove the effectiveness of a particular algorithm but to study the effect of users' behavior on a reputation system. We use a centralized system which assumes the correctness of information provided by users, e.g., it is not possible to give an evaluation of a user with whom there was no interaction. When verifying a file, the agents give a negative payoff to the agent uploading it, in case it's inauthentic. In turn, the system will spread it to the agents (if any) who uploaded it to the sender. There are two reputation thresholds: the first and higher one, under which it's impossible to ask for resources to other agents, the second, lower than the other, which makes it impossible even to share the owned files. This guarantees that an agents that falls under the first one (because she shared too many inauthentic files), can still regain credits by sharing authentic ones and come back over the first threshold. On the contrary, if she continues sharing inauthentic files, she will fall also under the second threshold, being de facto excluded from the network, still being a working link from and to other agents. The agents are randomly connected on a graph and feature the following parameters: Unique ID, Reputation value, set of neighbors, set of owned resources, set of goals (resources), set of resources being downloaded, set of suppliers (by resource). At each time step, agents reply to requests for download, perform requests (according to their goals) or verify files. While an upload is performed – if possible - each time another agent makes a request, requesting a resource and verification are performed in alternative. Verification ratio is a parameter for the simulation and acts stochastically on agents' behavior. All agents belong to two disjoint classes: malicious agents and loyal ones. They have different behaviors concerning uploading, while feature the same behavior about downloading and verification: malicious agents are simply agents who exploit for selfishness the weaknesses of the system, by always uploading inauthentic files if they don't own the authentic ones. Loyal agents, on the contrary, only upload a resource if they own it. A number of resources are introduced in the system at the beginning of the simulation, representing both the owned objects and the agents' goals. For coherence, an owned resource can't be a goal, for the same agent. The distribution of the resource is stochastic. During the simulation, other resources (and corresponding goals) are stochastically distributed among the agents. Each agent (metaphorically, the P2P client) keeps track of the providers, and this information is preserved also after the download is finished.

To test the limits and effectiveness of a reputation mechanism under different user behaviors an agent based simulation of a P2P network is used as methodology, employing reactive agents to model the users; these have a deterministic behavior based on the class they belong to (malicious or loyal) and a stochastic idealized behavior about verifying policy. Their use shows how the system works at an aggregate level. However, reactive agents can also be regarded as a limit for our approach, since real users have a flexible behavior and adapt themselves to what they observe. We built a model which is less idealized about the verifying factor, but it's still rigid when considering the agents' behavior about sending out inauthentic files. That's why we envision the necessity to employ cognitive agents based on reinforcement learning techniques. Though, reactive agents can also be a key point, in the sense that they allow the results to be easily readable and comparable among them, while the use of cognitive agents would have moved the focus from the evaluation of collaborative factor to that of real users' behavior when facing a RMS, which is very interesting, but beyond the purpose of the present work. In future works, this paradigm for agents will be considered.

The model is written in pure Java and does not make use of any agent development environment.

4. Model Specifications and Parameters

The P2P network is modeled as an undirected and non-reflexive graph. Each node is an agent, representing a P2P user. Agents are reactive: their behavior is thus determined a priori, and the strategies are the result of the stimuli coming from the environment and of the condition-action rules. Their behavior is illustrated in next section. Formally the multi agent system is defined as *MAS = <Ag; Rel>*, with *Ag* set of nodes and *Rel* set of edges. Each edge among two nodes is a link among the agents and is indicated by the tuple < *ai; aj* > with *ai* and *aj* belonging to *Ag*. Each agent features the following internal parameters:

– Unique ID (identifier),
– Reputation value (or credits) N(ai),
– Set of agent's neighbors RP(ai),
– Set of owned resources RO(ai),
– Set of goals (resource identifiers) RD(ai),
– Set of resources being downloaded P(ai),
– Set of pairs < supplier; resource >.

A resource is a tuple <*Name, Authenticity*>, where *Name* is the resource identifier and *Authenticity* is a Boolean attribute indicating whether the resource is authentic or not. The agent owning the resource, however, does not have access to this attribute unless he verifies the file.

The resources represent the object being shared on the P2P network. A number of resources are introduced in the system at the beginning of the simulation; they represent both the owned objects and the agents' goals. For coherence, an owned resource can't be a goal, for the same agent. The distribution of the resource is stochastic. During the simulation, other resources are stochastically introduced. In this way, each agent in the system has the same probabilities to own a resource, independently from her inner nature (malicious or loyal). In the same way also the corresponding new goals are distributed to the agents; the difference is that the distribution probability is constrained by its being possessed by an agent. Formally *R* be the set of all the resources in the system. We have that:
$\mathrm{RD}(a_i) \subseteq \mathrm{R}, \mathrm{RO}(a_i) \subseteq \mathrm{R}$ and $\mathrm{RD}(a_i) \cap \mathrm{RO}(a_i) = \emptyset$.
Each agent in the system features a set of neighbors *N(ai)*, containing all the agents to which she is directly linked in the graph: $\mathrm{N}(a_i) = \{a_j \in \mathrm{Ag} \mid < ai; aj > \in \mathrm{Rel}\}$. This information characterizes the information of each agent about the environment. The implemented protocol is a totally distributed one, so looking for the resource is heavily based on the set of neighbors.

In the real word the shared resources often have big dimensions; after finding the resource, a lot of time is usually required for the complete download. In order to simulate this the set of the "resources being downloaded" (*Ris*) introduced. These are described as *Ris = <resource ID, completion, check status>*, where *ID* is the resource identifier, *completion* is the percentage already downloaded and "*check status*" indicates whether the resource has been checked for authenticity or not. In particular, it can be not yet verified, verified and authentic and verified and inauthentic:

$\text{check status} \in \{\text{NOT CHECKED}; \text{AUTH}; \text{INAUTH}\}$

Another information is *ID* of the provider of a certain resource, identified by *P(ai)*. Each agent keeps track of those which are uploading to him, and this information is preserved also after the download is finished. The real P2P systems allow the same resource to be download in parallel from many providers, to improve the performance and to split the bandwidth load. This simplification should not affect the aggregate result of the simulation, since the negative payoff would reach more agents instead of just one (so the case with multiple provider is a sub-case of that with a single provider).

4.1. The Reputation Model

In this work we assume a simple idealized model of reputation, since the objective is not to prove the effectiveness of a particular reputation algorithm but to study the effect of users' behavior on a reputation system. We use a centralized system which assumes the correctness of information provided by users, e.g., it is not possible to give an evaluation of a user with whom there was no interaction. The reason is that we focus on the behavior of common agents and not on hackers who attack the system by manipulating the code of the peer application. In the system there are two reputation thresholds: the first and higher one, under which it's impossible to ask for resources to other agents, the second, lower than the other, which makes it impossible even to share the owned files. This guarantees that an agents that falls under the first one (because she shared too many inauthentic files), can still regain credits by sharing authentic ones and come back over the first threshold. On the contrary, if she continues sharing inauthentic files, she will fall also under the second threshold, being de facto excluded from the network, still being a working link from and to other agents.

4.2. The User Model

Peers are reactive agents replying to requests, performing requests or verifying files. While upload is performed each time another agent makes a request, requesting a file and verification are performed (in alternative) when it is the turn of the agent in the simulation. All agents belong to two disjoint classes: malicious agents and loyal agents. The classes have different behaviors concerning uploading, while they have the same behavior concerning downloading and verification: malicious agents are just common agents who exploit for selfishness the weaknesses of the system. When it is the turn of another peer, and he requests a file to the agent, he has to decide whether to comply with the request and to decide how to comply with it.

- The decision to upload a file is based on the reputation of the requester: if it is below the "replying threshold", the requestee denies the upload (even if the requestee is a malicious agent).

- The "*replyTo*" method refers to the reply each agent gives when asked for a resource. When the agent is faced with a request he cannot comply but the requester's reputation is above the "replying threshold", if he belongs to the malicious class, he has to decide whether to create and upload an inauthentic file by copying and renaming one of his other resources. The decision is based depending on a parameter. If the resource is owned, she sends it to the requesting agent, after verifying if her reputation is higher than the "replying threshold". Each agent performs at each round of simulation two steps:

1) Performing the downloads in progress. For each resource being downloaded, the agents check if the download is finished. If not, the system checks if the resource is still present in the provider's "sharing pool". In case it's no longer there, the download is stopped and is removed from the list of the "owned resources". Each file is formed by n units; when 2/n of the file has been downloaded, then the file gets automatically owned and shared also by the agent that is downloading it.

2) Making new requests to other peers or verifying the authenticity of a file downloaded or in downloading, but not both:

   a) When searching for a resource all the agents within a depth of 3 from the requesting one are considered. The list is ordered by reputation. A method is invoked on every agent with a reputation higher than the "requests threshold", until the resource is found or the list reaches the ending point. If the resource is found, it's put in the "downloading list", the goal is cancelled, the supplier is recorded and linked with that specific download in progress and her reputation is increased according to the value defined in the simulation parameters. If no resource is found, the goal is given up.

   b) Verification means that a file is previewed and if the content does not correspond to its description or filename, this fact is notified to the reputation system. Verification phase requires that at least one file must be in progress and it must be beyond the 2/n threshold described above. An agent has a given probability to verify instead of looking for a new file. In case the agent verifies, a random resource is selected among those “in download” and not checked. If authentic, the turn is over. Otherwise, a "punishment" method is invoked, the resource deleted from the "downloading" and from the "owned " lists and put among the "goals" once again.

The RMS is based on the "punishment" method which lowers the supplier's reputation, deletes her from the "providers" list in order to avoid cyclic punishment chains, and recursively invokes the "punishment" method on the punished provider. A punishment chain is thus created, reaching the creator of the inauthentic file, and all the aware or unaware agents that contributed in spreading it.

5. Results

The simulation goes on until at least one goal exists and/or a download is still in progress.

In the following table a summary of the most important parameters for the experiments are given:

Parameters	Value
Total number of agents	50
Total number of graph edges	80
Initial reputation (credits) for each agent	50
Percentage of loyal agents	50
Total number of resources at the beginning	50
Threshold to share owned resources	25
Threshold for requesting resources	10
Number of turns for introduction of new resources	1
Number of new resources to introduce	3
Total number of steps	2000

Table 1 – the main parameters

In all the experiments, the other relevant parameters are fixed, while the following ones change:

Parameters	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Positive payoff	1	1	1	1	1	1
Negative payoff	3	3	4	8	0	2
Verification percentage	30%	40%	30%	30%	30%	40%

Table 2 – the scenarios

A crucial index, defining the wellbeing of the P2P system, is the ratio among the number of inauthentic resources and the total number of files on the network. The total number is increasing more and more over time, since new resources are introduced iteratively. Another measure collected is the average reputation of loyal and malicious agents at the end of the simulation; in an ideal world, we expect malicious ones to be penalized for their behavior, and loyal ones to be rewarded. The results were obtained by a batch execution mode for the simulation. This executes 50 times the simulation with the same parameters, sampling the inauthentic/total ratio every 50 steps. This is to overcome the sampling effect; many variables in the simulation are stochastic, so this technique gives an high level of confidence for the produced results. In 2000 turns, we have a total of 40 samples. After all the executions are over, the average for each time step is calculated, and represented in a chart. In the same way, the grand average of the average reputations for loyal and malicious agents is calculated, and represented in a bar chart. In figure 1, the chart with the trend of inauthentic/total resources is represented for the results coming from experiments 1, 2, 3, 5 and 6. The results of experiment 4 is discussed later.

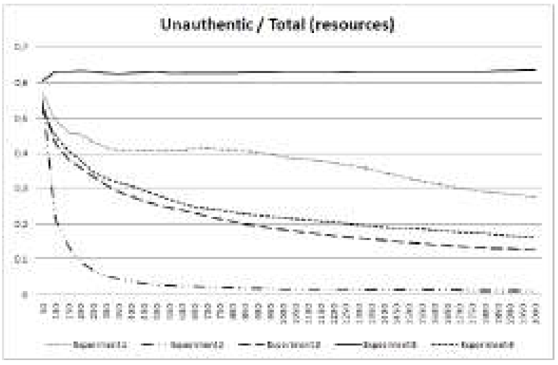

Figure 1 – inauthentic/total ratio

Experiment 5 depicts the worst case: no negative payoff is given: this is the case of a P2P network without a RMS behind it. The ratio initially grows and, at a certain point, it gets constant over time, since new resources are stochastically distributed among all the agents with the same probability. In this way also malicious agents have new resources to share, and they will send out inauthentic files only for those resources they do not own. In the idealized world modeled in this simulation, since agents are 50 malicious and 50 loyal, and since the ones with higher reputation are preferred when asking for a file, it's straightforward that malicious agents' reputation fly away, and that an high percentage of files in the system are inauthentic (about 63%). Experiment 1 shows how a simple RMS, with quite a light punishing factor (3) is already sufficient to lower the percentage of inauthentic files in the network over time. We can see a positive trend, reaching about 28% after 2000 time steps, which is an over 100% improvement compared to the situation in which there was no punishment for inauthentic files. In this experiment the verification percentage is at 30%. This is quite low, since it means that 70% of the files remain unchecked forever (downloaded, but never used). In order to show how much the human factor can influence the way in which a RMS works, in experiment 2 the verification percentage has been increased up to 40%, leaving the negative payoff still at 3. The result is surprisingly good: the inauthentic/total ratio is dramatically lowered after few turns (less than 10% after 200), reaching less than 1% after 2000 steps. Since 40% of files checked is quite a realistic percentage for a P2P user, this empirically proves that even the simple RMS proposed here dramatically helps in reducing the number of inauthentic files. In order to assign a quantitative weight to the human factor, in experiment 3, the negative payoff is moved from 3 to 4, while bringing back the verification percentage to 30%. Even with a higher punishing factor, the ratio is worse than in experiment 2, meaning that it's preferable to have a higher verification rate, compared to a higher negative payoff. Experiment 6 shows the opposite trend: the negative payoff is lighter (2), but the verification rate is again at 40%, as in experiment 2. The trend is very similar – just a bit worse - to that of experiment 3. In particular, the ratio of inauthentic files, after 2000 turns, is about 16%. At this point, it gets quite interesting to find the "*break even point*" among the punishing factor and the verification rate. After some empirical simulations, we have that, compared with 40% of verification and 3 negative payoff, if now verification is just at 30%, the negative payoff must be set to a whopping value of 8, in order to get a comparable trend in the ratio. This is done in experiment 4 (figure 2): after 2000 turns, there's 1% of inauthentic files with a negative payoff of 3 and a verification percentage of 40%, and about 0.7 with 8 and 30% respectively.

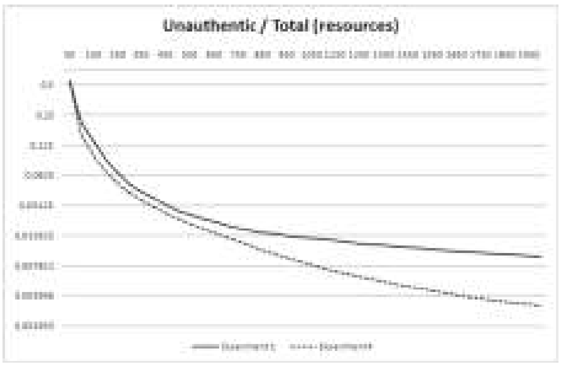

Figure 2 – weighting the collaboration factor

This clearly indicates that collaboration factor (the files verification) is crucial for a RMS to work correctly and give the desired aggregate results (few inauthentic files over a P2P network). In particular, a slightly higher verification rate (from 30% to 40%) weights about the same of a heavy upgrade of the punishing factor (from 3 to 8). This can be considered as a quantitative result, comparing the exogenous factor (resource verification performed by the users) to the endogenous one (negative payoff).

Besides considering the ratio of inauthentic files moving on a P2P network, it's also crucial to verify that the proposed RMS algorithm could punish the agents that maliciously share inauthentic files, without involving too much unwilling accomplices, which are loyal users that unconsciously spread the files created by the former ones. This is considered by looking at the average reputations, at the end of simulation steps (figure 3).

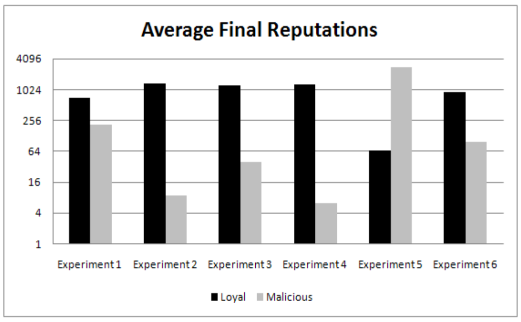

Figure 3 – final average reputations

In the worst case scenario, the malicious agents, that are not punished for producing inauthentic files, always upload the file they are asked for (be it authentic or not). In this way, they soon gain credits, topping the loyal ones. Since in the model the users with a higher reputation are preferred when asking files, this phenomenon soon triggers an explosive effects: loyal agents are marginalized, and never get asked for files. This results in a very low average reputation for loyal agents (around 70 after 2000 turns) and a very high average value for malicious agents (more than 2800) at the same time. In experiment 1 the basic RMS presented here, changes this result; even with a low negative payoff (3) the average reputations after 2000 turns, the results are clear: about 700 for loyal agents and slightly more than 200 for malicious ones. The algorithm preserves loyal agents, while punishing malicious ones. In experiment 2, with a higher verification percentage (human factor), we see a tremendous improvement for the effectiveness of the RMS algorithm. The average reputation for loyal agents, after 2000 steps, reaches almost 1400, while all the malicious agents go under the lower threshold (they can't either download or share resources), with an average reputation of less than 9 points. Experiment 3 explores the scenario in which the users just check 30% of the files they download, but the negative payoff is raised from 3 to 4. The final figure about average reputations is again very good. Loyal agents, after 2000 steps, averagely reach a reputation of over 1200, while malicious ones stay down at about 40. This again proves the proposed RMS system to be quite effective, though, with a low verification rate, not all the malicious agents get under the lower threshold, even if the negative payoff is 4. In experiment 6 the verification percentage is again at the more realistic 40%, while negative payoff is reduced to 2. Even with this low negative payoff, the results are good: most malicious agents fall under the lowest threshold, so they can't share files and they get an average reputation of about 100. Loyal agents behave very well and reach an average reputation of more than 900. Experiment 4 is the one in which we wanted to harshly penalize inauthentic file sharing (negative payoff is set at 8), while leaving an high laxity in the verification percentage (30%). Unlikely what it could have been expected, this setup does not punish too much loyal agents that, unwillingly, spread unchecked inauthentic files. After 2000 turns, all the malicious agents fall under the lowest threshold, and feature an average reputation of less than 7 points, while loyal agents fly at an average of almost 1300 points. The fact that no loyal agent falls under the "point of no return" (the lowest threshold) is probably due to the fact that they do not systematically share inauthentic files, while malicious agents do. Loyal ones just share the inauthentic resources they never check. Malicious agents, on the other side, always send out inauthentic files when asked for a resource they do not own, thus being hardly punished by the RMS, when the negative payoff is more than 3.

6. Whitewashing

A "*whitewashing*" mode is implemented and selectable before the simulation starts, in order to simulate the real behavior of some P2P users who, realizing that they cannot download anymore (since they have low credits or, in this case, bad reputation), disconnect their client, and then connect again, so to start from the initial pool of credits/reputation. When this mode is active, at the beginning of each turn all the agents that are under a given threshold reset it to the initial value, metaphorically representing the disconnection and reconnection. In experiments 7, 8 and 9 this is tested to see if it affects previous results.

In figure 4, the ratio among inauthentic and total resources is depicted, and in figure 5 the final average reputation for agents, when whitewashing mode is active.

Even with CBM activated, the results are very similar to those in which this mode is off. They are actually a bit worse when the negative payoff is low (3) and so is the verification percentage (30%): the ratio of inauthentic files in the network is quite high, at about 41% after 2000 turns versus the 27% observed in experiment 1, which had the same parameters, but no CBM. When the verification percentage is increased to 40%, though, things get quite better. Now the ratio of inauthentic files has the same levels as in experiment 2 (less than 1% after 2000 steps). Also with a lower verification percentage (again at 30%), but leaving the negative payoff at 4, the figure is almost identical to the one with the same parameters, but without a CBM. After 2000 turns, the inauthentic files ratio is about 12%.

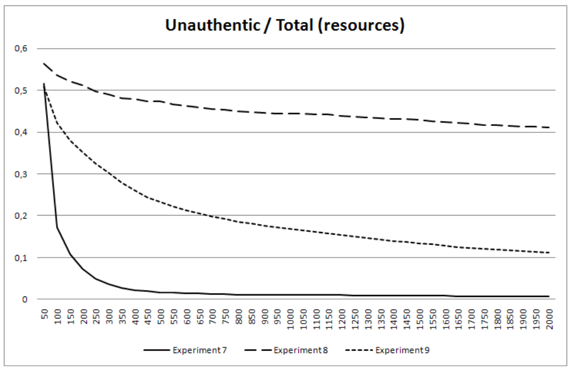

Figure 4 – inauthentic/total ratio in whitewashing mode

The experiments show that malicious agents, even resetting their own reputation after going below the lowest threshold, can't overcome this basic RMS, if they always produce inauthentic files. This happens because, even if they reset their reputation to the initial value, it's still low compared to the one reached by loyal agents; if they shared authentic files, this value would go up in few turns, but since they again start spreading inauthentic files, they almost immediately fall under the thresholds again.

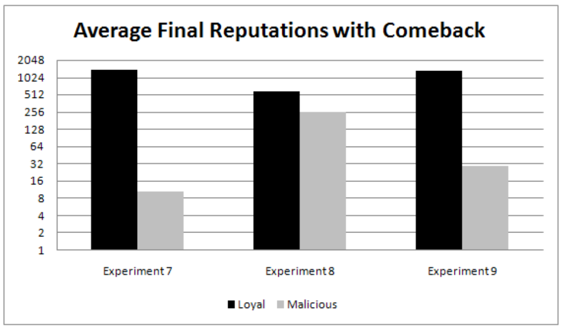

Figure 5 – final average reputations in whitewashing mode

7. Conclusion and Outlook

The main purpose of the work was to show, by means of an empirical analysis based on simulation, how the collaboration coming from the agents in a social system can be a crucial driver for the effectiveness of a RMS.

As a test-bed we considered a P2P network for file sharing and, by an agent based simulation, we show how a basic RMS can be effective to reduce inauthentic files circulating on the network. In order to enhance its performance, though, the collaboration factor, in the form of verifying policy, is crucial: a 33% more in verification results in about thirty times less inauthentic files on the network. While a qualitative analysis of this factor is straightforward for the presented model, we added a quantitative result, trying to weight the exogenous factor (the verification rate) by comparing it to the endogenous one (the negative payoff). We showed that a 33% increase in verification percentage leads to similar results obtained by increasing the negative payoff of 66%. Again, the collaboration factor proves to be crucial for the RMS to work efficiently.

While the provided results are encouraging, the model is not yet realistic under certain aspects. The weakest part is not the simplicity of the RMS algorithm or of the representation of the P2P network, rather the deterministic (reactive) behavior of the agents: the agents involved are too naive to represent real users. In particular, potentially malicious users try to exploit the weaker points of the system, by changing their behavior according to what they observe, like satisfaction of their own goals. It's very unlikely that users, when realizing not to download at the same rate as before, would go on sending out inauthentic files in the same way as before. Real users are flexible, and adapt themselves to different situations. If they see that many inauthentic files are moving on the network since informal norms regulating the P2P are not respected, it is likely that they would also start producing them, in order to gain credits, by an imitative behavior. While the use of reactive agents keeps the results more readable and easy comparable, in future works we'll implement cognitive ones, in order to explore their behavior under a RMS; they feature a policy which is dynamically created through trial and error, and progressive reinforcement learning. Two are the dimensions of learning that should be considered: one regarding the long term satisfaction of goals (related to the action of sending out an inauthentic file or not) and the other about the convenience in verifying a file (thus potentially losing a turn) related to the risk of being punished as an unwilling accomplice in spreading inauthentic files.

Besides, the threshold study now carried on at an aggregate level will be made also from the point of view of the individual agent: when does it become too costly to "cheat" for an agent so that it ceases to be beneficial? Such study will be made at a higher scale, referring to the number of agents and resources.

Also, if control through user collaboration has been studied, rewarding control should be considered as an individual incentive to control (with possible biases from malicious agent) and thus relate more to the collaboration objective of the study. This will also be studied in future works.

8. Acknowledgements

This work has been partially funded by the project *ICT4LAW*, financed by Regione Piemonte.

9. References

[1] A. Josang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. Decis. Support Syst., 43(2):618–644, March 2007.
[2] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. 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, 2003. ACM Press.