Cheat-Proof Trust Model for Cloud Computing Markets

Mario Macías and Jordi Guitart

Barcelona Supercomputing Center (BSC) and
Universitat Politecnica de Catalunya - Barcelona Tech (UPC)
Jordi Girona 29, 08034 Barcelona, Spain
{mario.macias,jordi.guitart}@bsc.es

Abstract. Online Reputation Systems would help mitigate the information asymmetry between clients and providers in Cloud Computing Markets. However, those systems raise two main drawbacks: the disagreement for assuming the cost of ownership of such services and their vulnerability to reputation attacks from dishonest parties that want to increase their reputation. This paper faces both problems by describing a decentralized model that would not need from the intervention of a central entity for managing it. This model includes mechanisms for allowing participants to avoid such dishonest behaviour from other peers: each client statistically analyses the external reports about providers and accordingly weights them in the overall trust calculation. The validity of the model is demonstrated through experiments for several use cases.

Keywords: Trust, reputation, cloud computing, dishonest behaviour.

1 Introduction

Online reputation systems help mitigate the information asymmetry between clients and providers in commerce markets. With the popularization of the World Wide Web, sites such as eBay [2] allow their users to submit and consult information about quality of products or the trustworthiness of both buyers and sellers. Such reputation systems enforce the confidence between parties and boost the number of commercial transactions.

This model has been also ported to Utility Computing Markets [14]. In Utility Computing Markets, both resource users and providers are autonomous agents that negotiate the terms of the Quality of Service (QoS) and the price that the client will pay to the provider for hosting their tasks or services in the resources. When the negotiation is finished, the terms of the contract are established in a Service Level Agreement (SLA). The most successful implementation of the Utility Computing paradigm is Cloud Computing, thanks to features of virtualization such as isolation of Virtual Machines (VMs), secure access to VMs with administrative privileges, or on-demand variation of the allocated resources.

Cloud Providers could not fulfil always the agreed QoS by several reasons, such as high load of resources, poor admission control, or dishonest behaviour.

K. Vanmechelen, J. Altmann, and O.F. Rana (Eds.): GECON 2012, LNCS 7714, pp. 154–168, 2012.

We suggest a reputation system to help clients choosing a provider and allow avoiding the providers with low QoS.

Traditional web reputation systems are based on reports from humans. This service can be part of a site (e.g. eBay reputation) or an independent site. They have clear business models: they increase the trust level to boost the economic transactions; also the service provider may get paid by advertisement. The incomes from the business model will amortize the cost of providing the service.

However, the aforementioned business model is not directly portable to Cloud Computing markets because the users and the providers of the resources are autonomous agents that are not able neither to communicate nor understand the human language; in addition, they are not a target for advertising campaigns and the Reputation Service Provider cannot make business from advertising. This raises two issues: (i) opinions about Cloud providers must be modelled for allowing their automatic processing; (ii) if there is no business model for a reputation service, nobody will provide it. There is many related work about modelling a reputation service (see section 2), but the need of making it economically feasible must be faced.

Reputation systems are vulnerable to reputation attacks [6]: dishonest companies can send biased opinions to increase their reputation or to decrease the reputation of their competitors. Such behaviour can be mitigated in traditional reputation systems by moderating the opinions. In addition, most users would be smart enough for discarding the dishonest reports. None of these methods can be applied to a decentralised, automatised agent-based reputation system.

Considering the aforementioned, these are our contributions:

- Description of a reputation model that applies to Cloud Computing business model and is easily implementable in a decentralized Peer-to-Peer (P2P) network. The cost of providing such service is not assumed by any central organisation; it is proportionally assumed by all the actors in the system.
- Statistical analysis for allowing market participants to detect dishonest behaviours from other peers that want to bias the true reputation of a provider.
- Validation of the model through experiments for several use cases.
- Discussion about the implementability of Reputation Systems in real Cloud Computing Markets.

The rest of the paper is structured as follows: after the presentation of the related work, section 3 describes the mathematical model created for describing the reputation of Cloud Computing providers. After the experimental validation of the model (section 4), section 5 discusses the requirements for implementing such system in a real Cloud market. At the end, we expose the conclusions and our future research lines.

2 Related Work

Our previous work [9] showed the importance of the reputation for a provider. To maintain a high reputation is a key factor for maximizing the revenue of

providers in Utility Computing Markets. We introduced a centralised proof-of-concept reputation architecture that relied in simple reputation models and ideal market conditions. This paper intends to be a step beyond: we add multiple reputation terms and a decentralized architecture which is robust to dishonest market actors.

This paper adopts some ideas from Azzedin et al. [4] and Alnemr et al. [3]: we differentiate between *direct* and *reputation* trust; we consider multiple provider facets to evaluate our trust methods; we also consider the trust factor to a recommender. Despite Azzedin et al. provide a reputation model, they do not detail how that would be implemented. This paper provides a pure mathematical model that is easily implementable for its computation. We detail and discuss some practical issues for implementing it in real platforms.

Rana et al. [15] monitor reputation from three points of view: Trusted Third Party, Trusted Module at Service Provider, and Model at Client Site. They introduce the figure of a trusted mediator to solve conflicts between parties. Our main objection is the difficulty to find some company or institution that is willing to host and maintain the trusted mediator, because the business model is not clear. In consequence, our paper suggest a purely P2P reputation mechanism.

This paper adopts various facets from the model of Xiong et al. [18] for ensuring the credibility of a feedback from a peer: number of transactions and transaction context. We agree with the necessity of a community-context factor for incentive peers for reporting true feedbacks. This paper differs from the work of Xiong et al. because we are focusing the particularities of current Cloud Computing markets: multiple SLOs, providers that are not integrated with the reputation system, and trust relations that are classified two types: trust on peers (for consultancy) and trust on providers (for commercial exchange).

Yu et al. [19] define a model in which reputation propagates through networks. They define a trust propagation operator that defines how trust propagates from a source peer (who reports the trust) to a destination peer in multiple steps. Unlike our paper, their model assumes the same trust both for service provision and trust report, and they do not update the trust on peers in function of the honesty of their reports.

The need of avoiding dishonest opinions in reputation systems is firstly raised by Kerr et al. [6]. In their work, the show several reputation attacks to allow dishonest peers to increase their revenue. They argue that the notion of 'security by obscurity' does not prevent attackers from cheating successfully. Our paper shows a method for protecting honest clients from dishonest peers that is complementary to other existing security mechanisms.

3 Description of the Reputation Model

3.1 Previous Definitions

Let $\overrightarrow{U} = (u_1, u_2, \ldots, u_n)$ and $\overrightarrow{V} = (v_1, v_2, \ldots, v_n)$ be two vectors that contain n elements. The Element-wise Product is defined as $\overrightarrow{U} \odot \overrightarrow{V} = (u_1 v_1, \ldots, u_n v_n)$ and the Element-wise Division is defined as $\overrightarrow{U} \oslash \overrightarrow{V} = (u_1/v_1, u_2/v_2, \ldots, u_n/v_n)$.

Let $CP = \{cp_1, cp_2, \ldots, cp_m\}$ be the set of m Cloud Providers that are competing in a market to sell their resources to the clients.

Let $C = \{c_1, c_2, \ldots, c_n\}$ be the set of n clients that want to host their services or tasks in the set CP of Cloud providers. Each client c_x is communicated to a set of peers, represented by the set $P_x = \{p_1^x, p_2^x, \ldots, p_r^x\}$, formed by r peers of client c_x. Each peer is also a client ($P_x \subseteq C$). The peer-direct communication between clients is established by means of Peer-to-Peer (P2P) networks [5].

When clients try to buy resources to host their services or applications, they send offers to the providers for starting a negotiation. Each provider owns a set of N physical machines. Each physical machine can host several VMs that execute single tasks, such as Web Services or Batch Jobs. The SLA of a task is described as $SLA = \{\overrightarrow{S}, \Delta t, Price\}$, in which $\overrightarrow{S} = (s_1, \ldots, s_k)$ are the Service Level Objectives (SLOs) that describe the amount of resources or the QoS terms to be purchased by the client. Each s_* term represents the number of CPUs, Memory, Disk, network bandwidth, and so on. Δt is the time period during which the task will be allocated in the VM. $Price$ is the amount of money that the client will pay to the provider for provisioning $\overrightarrow{S}$ at Δt.

Because this paper is about trust and not about revenue management, for simplification purposes this paper does not consider the direct economic penalties derived from the violation of the SLA terms. The details of revenue management by considering revenue and penalty functions can be referred in our previous work [10,8,11,13].

Both Cloud clients and providers are entities that have a degree of trust between them as individuals. The degree of trust can be expressed in multiple terms, represented as a Trust Vector: a client trusts a provider in multiple facets, related to the different terms of $\overrightarrow{S}$ (e.g. a Cloud provider could provide resources that are suitable for CPU-intensive applications but unstable in terms of network connection). Let $\overrightarrow{T}(A, B) = (t_1, \ldots, t_k)$ be the Trust Vector from the entity A to the entity B. This is, how much A trusts B. Both A and B belong to CP or C.

$\overrightarrow{T}(A, B) = \omega_1 \overrightarrow{D}(A, B) + \omega_2 \overrightarrow{R}(B)$; this is, the overall trust from A to B has two components: $\overrightarrow{D}(A, B)$ is the direct trust from A to B, which is built based on previous experiences between A and B; $\overrightarrow{R}(B)$ is the reputation trust, which is calculated by asking the set of peers of entity A about their experiences with B (see section 3.2, equation 2) . In plain words, the direct trust is *what A directly knows about B* and the reputation trust is *what the others say about B*. ω_1 and ω_2 are used to weight how much importance the client assign to each of the terms, and may vary in function of each particular client. All the terms of $\overrightarrow{T}$, $\overrightarrow{D}$ and $\overrightarrow{R}$ are real numbers between 0 (no trust) and 1 (maximum trust).

Because trust and reputation have many terms, a provider could deserve high trust when considering some SLOs and low trust when considering others. This does not have to be detrimental to a given client. For example, a provider that deserves high trust only in terms of CPU could not be suitable for many applications such as web services or databases, but could be suitable for some

CPU-intensive scientific applications. Some types of workloads can be allocated in such providers with a high degree of trustworthiness. This raises a question: which incentive would clients have for allocating their workloads in such providers? Would it not be better to allocate them in providers whose trust level is high in all the terms of $\overrightarrow{T}(A, B)$? The response would be affirmative if there were not economic incentives at client side. Previous work from the authors [10,8,11] shown the economic benefit for both clients and providers of dynamically negotiating the prices in function of many factors, such as offer/demand ratio, allocated resources or QoS, and how those prices could vary in function of the reputation of the provider [9]. If a provider is able to guarantee the QoS requirements of a client at lower prices, the client will have incentive to allocate there its workloads; even if the provider has low reputation in factors that are not important for the client.

Considering the aforementioned, each client c_x has its own Trust Ponder Vector $\overrightarrow{I}(c_x)$, which weights each of the SLOs of $\overrightarrow{T}(A, B)$ in function of the importance the client assigns to each of them. The Element-Wise product $\overrightarrow{T}(c_x, cp_y) \odot \overrightarrow{I}(c_x)$ returns a vector that scores how trustworthy is the provider p_y in function of three facets: the reputation of cp_y, the direct trust from c_x to cp_y and the QoS needs of c_x. All the terms of $\overrightarrow{I}$ are real numbers between 0 and 1.

Let $Score(SLA, c_x, cp_y)$ be a function that scores the suitability of the provider cp_y in function of the SLA and the trust from client c_x to provider cp_y. For each SLA negotiation, the client will choose the provider whose $Score$ is the highest.

The definition of $Score_y^x$ may vary depending on the client policies and negotiation strategies. For evaluating the validity of the model, the clients evaluated in this paper score the providers according to equation 1. In this equation, the scores are always negative. The nearer to 0 the better score. The client divides the calculated trust from c_x to cp_y by the Trust Ponder Vector (element-wise division), and the negative of the magnitude of the resulting vector gives a scoring that shows how trustworthy is a provider for the preferences of c_x (in positive it would be the lower the better, that is why the result is multiplied by -1). This score is divided by the price: the client would accept sending tasks to providers to which the trust is lower if the price they establish is low enough.

$$Score(SLA, c_x, cp_y) = -\frac{\left\| \overrightarrow{T}(c_x, cp_y) \oslash \overrightarrow{I}(c_x) \right\|}{Price} \tag{1}$$

The scoring function in equation 1 will incentive providers to keep its maximum trust level and, if not possible, to lower prices.

3.2 Dishonest Behaviour towards the Reputation Model

A Cloud Provider could not provide the amount of resources that previously agreed with a given client. This fact can be caused by technical failures [17], errors in the calculation of the number of resources to provide [8,11], or dishonest behaviour. The reputation model described in this section is intended to alert the market participants when a provider is not fulfilling its agreed SLAs.

However, dishonest providers could enable fake clients to perform collusion: to report false or dishonest feedback for (1) increasing artificially the reputation of a provider; or (2) decreasing artificially the reputation of other providers from the competition. Since our reputation model is decentralized and unmanaged, the clients need a model for preventing false reports from dishonest peers.

Let $T(c_x, p_y)$ be a single-term trust relation from a client $c_x \in C$ to one of its peers $p_y \in P_x$. Let $P_x^z = \{p_1^z, \ldots, p_s^z\} \subseteq P_x$ the subset of s peers of c_x that have any direct trust relation to provider cp_z (this is, they can report previous experiences to cp_z), the Reputation Trust from c_x to cp_z is calculated as:

$$\overrightarrow{R}(c_x, cp_z) = \sum_{y=1}^{S} \left(T(c_x, p_y^z) \cdot \overrightarrow{D}(p_y^z, cp_z) \right) \oslash \sum_{y=1}^{S} \overrightarrow{T}(c_x, p_y^z) \tag{2}$$

Equation 2 is calculated by asking the peers that have any direct relation with cp_z and pondering their reports by the direct trust from the client to its peers. The report of a client to which there is high trust has more weight than the report of a client to which there is low trust. The key issue is to establish this trust relation between a client and its peers to avoid dishonest behaviours and give more consideration to the accurate reports.

The trust relation between a client and its peers is continuously updated in base to the next assumption: most peers are honest and, when asked, they report their true validation to the provider. Related work considers many incentives to peers for reporting honestly [20,7]. Our contribution is complimentary to them, since we deal with the minimization of the impact of the dishonest reports.

Assuming the aforementioned, the trust from a client to each of its peers is calculated according to algorithm 1:

begin
  The average values and the variances of all the reports from the peers of the P_x^z set are stored, respectively, in $\overrightarrow{A}$ and $\overrightarrow{\Sigma^2} = (\sigma_1^2, \ldots, \sigma_s^2)$;
  foreach p_y^z *in* P_x^z **do**
    $\overrightarrow{F} \leftarrow \overrightarrow{A} - \overrightarrow{D}(p_y^z, cp_z) = (a_1 - d_1, \ldots, a_s - d_s)$;
    foreach $|a_n - d_n|$ *in* $\overrightarrow{F}$ **do**
      if $|a_n - d_n| > \alpha \cdot \sigma_n^2$ **then**
        Decrease $T(c_x, p_y)$;
      else
        Increase $T(c_x, p_y)$;
      end
    end
  end
end

Algorithm 1. Updating trust from c_x to all its peers

To detect *potentially* bad reputations, algorithm 1 checks which peers reported a trust which is far from the other reports for the same provider. We stress *potentially* because, by any reason, a honest peer could have been provided with bad QoS while the others do not: because a punctual failure, or because the provider starts to underprovision QoS by an outage or because it starts to behave dishonestly when its reputation is high enough. These cases must not penalise too much the client that starts reporting different than the others. Only repetitive reports that are different would decrease considerably the reputation of a client.

There are two parts of algorithm 1 that will depend on the client policies. α multiplies the variance of the trust reports, and indicates how tolerant is the client with the concrete reports that are far from the average. The lower α, the lower tolerance. The other part that depends on the client policy is the function to increase or decrease the trust on a peer. In this paper we have used a piecewise-defined function that multiplies $T(c_x, p_y)$ in function of how far the trust report from the average. If there is no difference from a report to the average of all the other reports, the trust relation is multiplied by $MAX\_REWARD > 1$. The trust relation is not affected when $|a_n - d_n| = \alpha \cdot \sigma_n^2$, and if $|a_n - d_n| > \alpha \cdot \sigma_n^2$, the trust relation is multiplied to a minimum of $MAX\_PENALTY < 1$. Instead of the simplicity of $f(x)$, it is proven as effective in the evaluation (section 4).

Figure 1 shows that the slope of the linear function that penalizes the trust is less pronounced than the slope of the linear function that rewards the trust. In addition, $\frac{MAX\_PENALTY+MAX\_REWARD}{2} < 1$. The reasons are two: (1) the imbalance between $MAX\_PENALTY$ and MAX_REWARD will difficult that dishonest peers recover easily their trust; and (2) honest peers that, by any reason, punctually report values near $\alpha \cdot \sigma_n^2$ are not penalized with severity. Previous experimentations demonstrated that not dividing the function in pieces with different slopes would entail too much instability in the trust updating, and honest peers would lose their trust without solid reasons.

When the trust to a peer reaches 0, it is definitely expelled from the trust ring of the client, and its trust cannot be recovered any more.

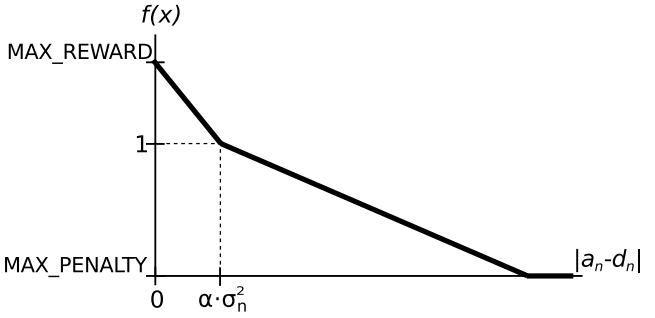

Fig. 1. Function to multiply the trust to a given peer, based on its previous report

3.3 Considering Reputation at SLA Negotiation Stage

As proven in previous work [9], low reputation lead to decrease the revenue of a Grid Provider: the lower trust the less currency will the clients pay for a service. In other words, if two providers offer the same QoS at equal prices, the client will choose the provider whose reputation is the highest. By this reason, a provider needs to adjust its price to its real reputation due to the effects of market competition. Our previous work showed how prices may be dynamically decided in function of many facets, such as number of resources, QoS, client relationship and market status [8,10,11,12,13]. This paper also defends the need of considering reputation as an additional facet when the provider negotiates a SLA with the client. There are two reasons: adjusting the revenue to the reputation will allow providers to maximize their benefit when reputation is high and sell its resources when reputation is low; the other reason is that selling the resources when reputation is low will allow provider recover its reputation.

Pricing in function of the trust involves two key issues that must be solved:

Calculating the trust from a given client. As seen in section 3.1, the trust from a client to a provider depends on three factors: the direct trust, the reputation as reported by all peers, the Trust Ponder Vector, and the weights that a particular client assigns to both direct trust and reputation. Direct Trust and Reputation can be approximated statistically, but the Trust Ponder Vector and the weights are completely private parameters that depend on the preferences of the client.

Defining a pricing function. Each provider must decide what are the proportion and distribution that trust would influence the prices. It is difficult to model because it depends on the emergent behaviour of all the market clients. Our previous work demonstrated that Genetic Algorithms [12] are suitable for this type of problems, because they rapidly adapt the pricing function to a changing/unknown environment. However, for simplification purposes, this simulation in this paper uses linear correlations between reputation and price [9].

4 Experiments

This section validates the model in section 3 by means of a custom Market Reputation Simulator [1]. In the simulation, clients look for resources for allocating their tasks in the providers that fit their QoS requirements. The experiments consider three SLOs: CPU, disk and network bandwidth. Therefore the Trust Vector and the Trust Ponder Vector is formed by 3 terms. Each experiment is a succession of market iterations. Each market iteration performs the next steps for all the clients in the market:

1. The client sends an offer to the providers. The offer specifies the QoS requirements and the time slot. The providers that have enough resources to handle it return a price.
2. The client asks its peers for the reputation of the providers that returned a price.

3. The client scores all the providers according to equation 1. It reaches an agreement with the provider whose score is the highest.
4. The client updates its trust to its peers according to equation 2. When the task is executed, it also updated its direct trust relation to the provider in function of the actual QoS.

The simulations rely on some constant values of which functionality is not to reflect real market data, but to evaluate the model in terms of relative results and tendencies: the honest providers whose resources work normally provide around 97% of the agreed QoS; at the beginning of the experiments, all providers and clients have an initial direct trust of 0.5. Other constant values are described in their respective experiments.

4.1 Basic Provider-Side Reputation

In the first experiment, five providers are competing in a market during 100 simulation steps. Four providers are honest and a provider is behaving dishonestly: it only provides the 60% of the QoS that it has previously agreed with the client. In addition, one of the honest providers suffers an outage [17] in its network at step 33. In consequence, it is providing the 50% of its network capacity until step 67.

Figure 2 shows the average trust from the clients to the providers. All the elements of the trust vectors are shown separately, but grouped the next way: the trust terms corresponding to the SLOs of the dishonest provider are shown as crosses; the trust element corresponding to the network of the provider that suffers the outage is a continuous line; the trusts for the rest of SLOs are shown as points. Figure 2 shows that the dishonest provider has a reputation proportional to the percentage of agreed QoS that is providing. The market also quickly notices that one of the providers is starting to provide a bad QoS in network and, after a quick decrease of the reputation, it slowly converges to 0.5, which corresponds to the percentage of QoS that is providing due to the outage. When the provider solves its network problems, its reputation increases fast, until it converges to the average reputation of the other SLOs.

4.2 Client-Side Reputation

This section evaluates the trust relations between peers in the scenario of the previous section. In that experiment, the market demand is formed by 24 clients that negotiate with the providers for allocating the workloads in the cloud resources. Before starting a negotiation with a provider, a client ask its peers for the reputation of the provider, then weight it with its direct trust (if any) and multiply it by the Trust Ponder Vector $\overrightarrow{I}(c_x)$. When the provider returns a price for a requested amount of resources, the client evaluates it in function of the price and the pondered trust.

When the client calculates the reputation of a provider, it tries to detect the dishonest peers as explained in section 1: it decreases or increases its trust to

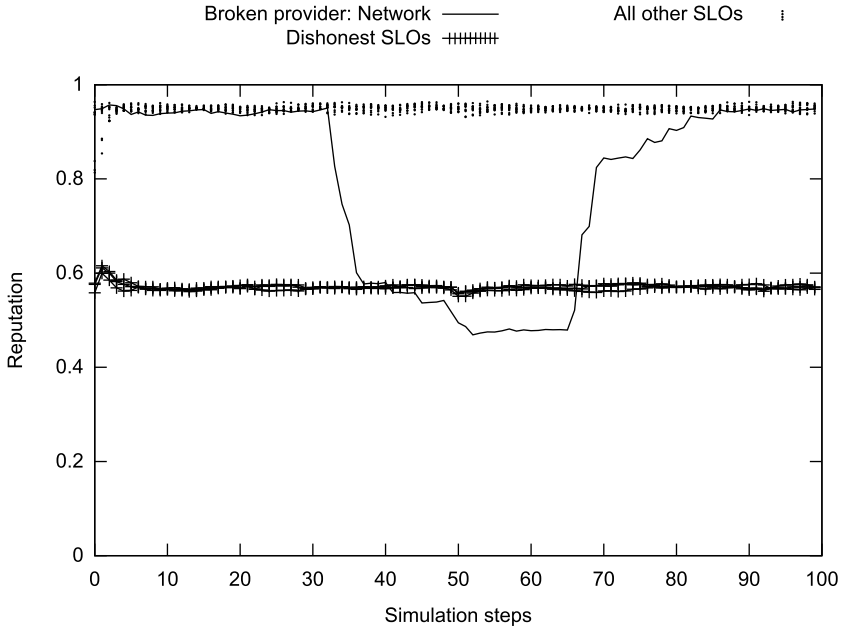

Fig. 2. Evaluation of reputation of providers

each peer in function of what they report. This paper does not intend to set the optimum values for MAX_REWARD and $MAX\_PENALTY$ constants (figure 1), so we have set $MAX\_REWARD = 1.05$ and $MAX\_PENALTY = 0.8$ as intuitive values for showing the tendencies. Different values would make the trust to peers evolve quicker or slower.

In the experiment, the dishonest provider infiltrated two peers that report trust values near 1 for the dishonest provider (while its real reputation is 0.5) and the 50% of the actual trust for the other providers. Figure 3 shows that, as initial state, all the peers of a given client have a trust of 0.5. The first dishonest client is reporting false trust values from the beginning, so it is quickly expelled from the list of peers (when it reaches trust 0). The trust to all the other providers is increased, including the second dishonest peer, whose strategy is to increase its reputation for increasing the influence of its false trust reports in the future. When the second dishonest client starts cheating at step 50, the client detects it and progressively decreases its reputation until reaching 0 value at step 59.

4.3 Effectiveness of Scoring Function for Allocating Tasks

To evaluate the effectiveness of equation 1 as rule for selecting a suitable provider while saving money, four providers are competing in a market for selling CPU, Disk and Network Bandwidth as SLOs: the first provider has the maximum reputation in all the SLOs; the second, third and fourth provider have the maximum reputation in all the SLOs but in CPU, Disk and Network, respectively. 32 clients want to submit their workloads to the providers, so they score them in

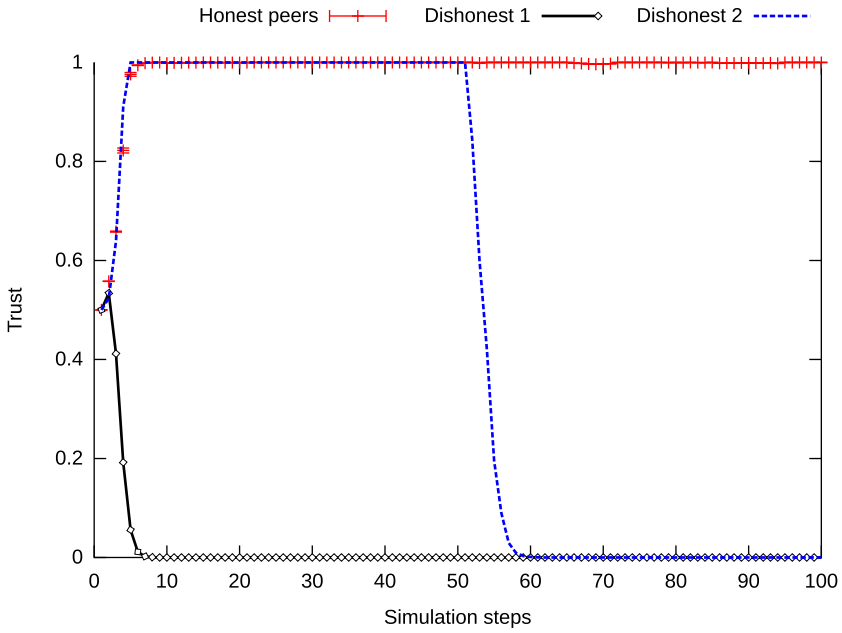

Fig. 3. Evaluation of trust to peers

Table 1. Values of the Trust Ponder Vector for each group of clients

Group	$\overrightarrow{I}(c_x) = (i_{cpu}, i_{disk}, i_{network})$
1	$(1, 1, 1)$
2	$(1, 0.3, 0.3)$
3	$(0.2, 0.8, 0.6)$
4	$(0.6, 0.3, 1)$

function of the trust, the Ponder Vector, and the price they ask. The 32 clients are divided in four groups depending on which necessities they have respecting the trust to each SLO (see table 1). Values of table would correspond to different types of workloads, for example: applications with a balanced resource usage (group 1), CPU-intensive applications that do neither intensively use disk nor network (group 2), Database applications with intensive disk and network usage (group 3) or some kind of web services that intensively use CPU and Network but not disk (group 4). The values of table 1 do not reflect any real measure of workloads. Their purpose is to be varied to see how the scoring function of equation 1 behaves.

In the first few iterations of the experiment, the tasks are allocated in the different providers pseudo-randomly. When the reputation of each provider is near to their true QoS, the next allocation of tasks is measured:

– All tasks from group 1 are placed in the provider with maximum QoS in all the SLOs. QoS is critical for this group and they are not willing to allocate their tasks in other providers despite the lower prices.

- All tasks from group 2 are placed at ∼50% in provider with low network reputation and ∼50% in provider with low disk reputation. Only CPU is critical for this group.
- All tasks from group 3 are placed in providers with low CPU reputation, because other SLOs have high importance.
- All tasks from group 4 are placed in provider with low disk reputation, because disk is the SLOs with the lowest importance.

The measured results tend to round numbers (e.g. 100% of tasks are allocated in the same provider when the system becomes stable) because of the experiment is repeated in a controlled simulation environment. A real market would add some statistical noise to the results.

5 Discussion: Implementing the Model in a Real Market

This paper demonstrates the validity of the reputation model from an experimental point of view. Since this paper is focused on the definition of the model, some implementation details are not considered from a formal view. This section wants to argue the implementability of the model, and what are the conditions for allowing the reputation model being feasible from the trust and economic side. Summarizing, we identify the next requirements:

- It is required to specify a communication protocol about trust information exchange for all the peers in the same network.
- A digitally-signed proof of purchase must be provided by peers that report their trust to a provider. The proof of purchase could be the agreed SLA, digitally signed by both client and provider. In consequence, a trustworthy Cloud Market requires certification authorities and identity management.
- Precisely quantify the SLAs to measure whether the provider is allocating all the resources to fulfil them. Some resources, such as CPU cycles, are difficult to measure accurately from a client side. We suggest negotiating in terms of high-level metrics (e.g. web-services throughput) and then translate such high-level metrics to low-level metrics by means of SLA decomposition [16].

The cost of implementing our trust model is not carried out by any centralised component, but it is shared by all the peers. The cost for each peer, in terms of memory space and extra calculations, is the next: let s the number of SLOs in a SLA; let r be the number of peers of a client; let m be the number of Cloud Providers. According to the model of section 3.1, the complexity of calculating the trust of all the providers is $O(s \cdot m \cdot r)$. According to algorithm 1, the complexity of updating the trust from a client to all its peers is $O(r \cdot s)$.

In terms of space complexity, a client needs to store a $O(m \cdot s)$ map with all the direct trust values to all the providers, and another $O(r)$ map with all the direct trust values to its peers.

The incentive-compatibility property of the mechanism must also be discussed. We suggest Cloud providers to penalize dishonest peers by increasing

the price of their resources for such type of peers. This has two positive effects on the market: peers are encouraged to report the true valuation of the service providers, and providers get an economic compensation for possible reputation attacks, as if it were an assurance.

6 Conclusions and Future work

This paper describes a reputation model that faces some open issues in the state of the art. First, we propose a P2P architecture for dealing with the cost of provision of centralized reputation services, which may be a good architecture for other markets but not for Cloud Computing. Second, we define a mathematical model for calculating the trust relationship from a client to a provider. This model also defines trust relations between peers and updates them in function to statistical analysis for detecting the trustworthiness of their reports. The validity of the model is demonstrated through exhaustive experiments in three use cases: calculation of the trust in a scenario with a dishonest provider and a provider that suffers an outage; calculation of the trust between peers in a scenario with dishonest clients that report false data about providers; usage of the model for the economic benefit of the clients in function of their requirements.

This paper opens a wide range of future work lines: the model can be used also by providers to improve their business models. By evaluating trust, they can analyse the economic consequences of their resource management policies (for example, to calculate the impact in reputation of cancelling a task from a given client [11,13]). The trust information may also be used for allowing more accurate negotiations with clients. This requires opening another research line: how to statistically poll and evaluate the reputation of a provider in the market for reducing the uncertainty.

Another future work line is to improve the model at trust level for avoiding other types of reputation attacks, such as coordinated attacks or whitewashing (reporting well on small transactions for acquiring high reputation and then attack for high price contracts, and then disappear).

Acknowledgements. This work is supported by the Ministry of Science and Technology of Spain and the European Union (FEDER funds) under contract TIN2007-60625, by the Generalitat de Catalunya under contract 2009-SGR-980, and by the European Commission under FP7-ICT-2009-5 contract 257115 (OPTIMIS).

References

1. Market Reputation Simulator, `https://github.com/mariomac/reputation`
2. eBay (2012), `http://www.ebay.com/`
3. Alnemr, R., Koenig, S., Eymann, T., Meinel, C.: Enabling usage control through reputation objects: A discussion on e-commerce and the internet of services environments. Journal of Theoretical and Applied Electronic Commerce Research 5(2), 59–76 (2010)

4. Azzedin, F., Maheswaran, M.: Evolving and managing trust in grid computing systems. In: Proceedings of the IEEE Canadian Conference on Electrical Computer Engineering CCECE 2002, Winnipeg, Manitoba, Canada, vol. 3, pp. 1424–1429 (2002)
5. Gupta, M., Judge, P., Ammar, M.: A reputation system for peer-to-peer networks. In: 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2003), pp. 144–152. ACM, Monterey (2003)
6. Kerr, R., Cohen, R.: Smart cheaters do prosper: defeating trust and reputation systems. In: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, AAMAS 2009, Richland, SC, vol. 2, pp. 993–1000 (2009), http://dl.acm.org/citation.cfm?id=1558109.1558151
7. Kerr, R., Cohen, R.: Trust as a tradable commodity: A foundation for safe electronic marketplaces. Computational Intelligence 26(2) (2010)
8. Macias, M., Fito, O., Guitart, J.: Rule-based SLA management for revenue maximisation in cloud computing markets. In: 2010 Intl. Conf. of Network and Service Management (CNSM 2010), Niagara Falls, Canada, pp. 354–357 (October 2010)
9. Macias, M., Guitart, J.: Influence of reputation in revenue of grid service providers. In: 2nd International Workshop on High Performance Grid Middleware (HiPerGRID 2008), Bucharest, Romania (November 2008)
10. Macias, M., Guitart, J.: Using resource-level information into nonadditive negotiation models for cloud market environments. In: 12th IEEE/IFIP Network Operations and Management Symposium (NOMS 2010), Osaka, Japan, pp. 325–332 (April 2010)
11. Macías, M., Guitart, J.: Client Classification Policies for SLA Negotiation and Allocation in Shared Cloud Datacenters. In: Vanmechelen, K., Altmann, J., Rana, O.F. (eds.) GECON 2011. LNCS, vol. 7150, pp. 90–104. Springer, Heidelberg (2012)
12. Macías, M., Guitart, J.: A genetic model for pricing in cloud computing markets. In: Proceedings of the 2011 ACM Symposium on Applied Computing, SAC 2011, TaiChung, Taiwan, pp. 113–118 (2011), http://doi.acm.org/10.1145/1982185.1982216
13. Macias, M., Guitart, J.: Client classification policies for SLA enforcement in shared cloud datacenters. In: 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2012), Ottawa, Canada, pp. 156–163 (May 2012)
14. Neumann, D., Stoesser, J., Anandasivam, A., Borissov, N.: SORMA – Building an Open Grid Market for Grid Resource Allocation. In: Veit, D.J., Altmann, J. (eds.) GECON 2007. LNCS, vol. 4685, pp. 194–200. Springer, Heidelberg (2007)
15. Rana, O., Warnier, M., Quillinan, T.B., Brazier, F.: Monitoring and Reputation Mechanisms for Service Level Agreements. In: Altmann, J., Neumann, D., Fahringer, T. (eds.) GECON 2008. LNCS, vol. 5206, pp. 125–139. Springer, Heidelberg (2008)
16. Reig, G., Alonso, J., Guitart, J.: Prediction of job resource requirements for deadline schedulers to manage high-level SLAs on the cloud. In: 9th IEEE Intl. Symp. on Network Computing and Applications, Cambridge, MA, USA, pp. 162–167 (July 2010)
17. Tehrani, R.: Amazon EC2 outage: what the experts tell us. Customer Interaction Solutions 29(12), 1 (2011)
18. Xiong, L., Liu, L.: Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. IEEE Transactions on Knowledge and Data Engineering 16(7), 843–857 (2004)

19. Yu, B., Singh, M.P.: A Social Mechanism of Reputation Management in Electronic Communities. In: Klusch, M., Kerschberg, L. (eds.) CIA 2000. LNCS (LNAI), vol. 1860, pp. 154–165. Springer, Heidelberg (2000)
20. Zhang, J.: Promoting Honesty in Electronic Marketplaces: Combining Trust Modeling and Incentive Mechanism Design. Ph.D. thesis, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada (May 2009), http://hdl.handle.net/10012/4413