Analysis of a Trust Model for SLA Negotiation and Enforcement in Cloud 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

December 21, 2015

Abstract

Online Reputation Systems 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 article faces both problems by describing a decentralised (peer-to-peer) trust model that does not require the intervention of a central entity to manage it. This model includes mechanisms to allow participants to avoid dishonest behaviour from other peers: each client statistically analyses the external reports about providers and updates the trustworthiness of the peers. The trustworthiness values will be used to negotiate prices in later transactions. The trust model is then incorporated in the Service-Level Agreement negotiation and enforcement processes, prioritising trusted clients over non-trusted clients to minimise the consequences of low Quality of Service in relation to the trust of the provider, and incentivise accurate trust reports from the clients. Finally, this article evaluates and discusses the validity of the trust model under different attacks from dishonest clients and providers.

1 Introduction

Online reputation systems help mitigate the information asymmetry between clients and providers in the markets. With the popularisation of the World Wide Web, sites such as eBay [2] allow their users to submit and consult information about the 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 [4]. 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 [14]. 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 virtualisation such as isolation of Virtual Machines (VMs), secure access to VMs with administrative privileges, and on-demand variation of the allocated resources.

Cloud Providers can violate the agreed SLAs by several reasons, such as high load of resources, poor admission control, or dishonest behaviour. We suggest the use of a reputation system to help clients choose a provider and allow them to avoid 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 amortise the cost of providing the reputation 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 neither able to communicate nor understand the human language; in addition, they are not a target for advertising campaigns and the provider of the reputation service cannot make business from advertising. This raises two issues: (i) opinions about Cloud providers must be modelled to allow their automatic processing; (ii) if there is no business model for a reputation service, nobody will provide it. There is much 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 [10]: 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 to discard the dishonest reports. None of these methods can be applied to a decentralised, automated agent-based reputation system.

Reputation allows markets to exclude dishonest providers. However, spot failures or system outages may also decrease the reputation of honest providers. These outages have a double economic impact: the provider must pay penalties for the VMs whose QoS has not been fulfilled, and he will lose future clients due to the loss of reputation. For this reason, providers operating in a Cloud Computing market require trust-aware management policies aimed at retaining their reputation when unexpected failures occur.

The main contributions of this work are as follows:

- Definition of a trust model that applies to a Cloud Computing business model and is easily implementable in a decentralised Peer-to-Peer (P2P)

network [7]. Our P2P configuration avoids the intervention of any central authority or super-peers, which are a potential source of manipulation of the reports. Peers ask for information about the providers by means of a query flooding variation that is optimised with probabilistic routing tables to enhance its scalability [12]. The trust model is composed by two facets: direct trust, based in previous experiences between a client and a provider; and reputation trust, based in the information that the peers of a client provide about a provider. The cost of providing such service is not assumed by any central organisation; it is proportionally assumed by all the actors in the system. The model uses statistical analysis to allow market participants to detect dishonest behaviours from other peers that want to bias the true reputation of a provider.

- Evaluation of the operation of Cloud infrastructures that considers the impact of the reputation in the revenue and definition of policies to minimise the impact of system failures in the reputation. On one side, we enable the provider to discriminate clients according to their reputation to favour those with high reputation under some conflicting situations, since those clients will impact positively on its reputation. On the other side, the provider analyses the impact of management actions in its reputation and its revenue every time it has to make a decision.

- Evaluation and discussion of the vulnerability of the reputation system toward different attacks: self-promoting, slandering, orchestrated attacks, whitewashing, and denial of service.

The rest of the article 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. Section 4 proposes a set of policies to allow providers considering their reputation during the negotiation and enforcement of SLAs. After the experimental validation of the model from both clients and provider side in Section 5, Section 6 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 [16] showed the importance of the reputation for a provider. Maintaining a high reputation is a key factor to maximise the revenue of providers in Utility Computing Markets. We introduced a centralised proof-of-concept reputation architecture that relied on simple reputation models and ideal market conditions. This article intends to go a step further: we add multiple reputation terms and a decentralised architecture that is robust to dishonest market actors.

This article adopts some ideas from Azzedin et al. [5] and Alnemr et al. [3]: we differentiate between *direct* trust and *reputation* trust; we consider multiple provider facets to evaluate our trust methods; we also consider the trust factor

to a recommender. Azzedin et al. provide a reputation model, but they do not detail how that would be implemented. This article provides a pure mathematical model that is easily computable. We detail and discuss some practical issues to implement it in real platforms.

Rana et al. [22] 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. The main issue with their approach 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. Therefore, our article suggest a purely P2P reputation mechanism.

This article adopts various facets from the model of Xiong et al. [26] to ensure the credibility of a feedback from a peer: number of transactions and transaction context. We agree with the necessity of a community-context factor to incentivise peers for reporting true feedbacks. This article 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 in two types: trust on peers (for consultancy) and trust on providers (for commercial exchange).

Yu et al. [27] 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 article, 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 was firstly raised by Kerr et al. [10]. In their work, they 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 article shows a method for protecting honest clients from dishonest peers that is complementary to other existing security mechanisms.

Our article discusses the need to motivate users to perform true reports about the QoS of the providers. There are many proposals to motivate users to report true validations, e.g. micro payments as reward for the true reports [21, 9]. These incentives are part of the reputation system. Such central entity that pays clients is not feasible in a decentralised, peer-to-peer reputation system. In our work, the incentives are enforced by the actors of such system: clients may discriminate providers or peers that are behaving dishonestly.

3 Description of the reputation model

In our reputation model, every time a client wants to acquire information about the reputation of a provider, it must query the rest of peers in the P2P network and the peers that have previous experiences about it. The aim of this paper is not to define and implement in detail a P2P architecture, but we propose the

following aspects to define our P2P architecture:

Routing. P2P networks must implement some form of virtual overlay network [13]. Such overlay networks can be *structured* or *unstructured*. In structured P2P networks, the overlay is organized into a specific topology (e.g. a tree). They usually implement a distributed hash table (DHT) which stores key/pair values that associate a given information with the peer that owns it. Looking for the information of a Cloud Provider with DHTs is efficient, but DHTs are not robust in networks where peers frequently join and leave the network. Unstructured networks, which are built over randomised topologies, are easy to build and highly robust to networks with high rate of peers that join and leave the network. This paper suggests an **unstructured** topology, since it does not require the intervention of any central tracker or super-peer, which would be a point for potential attacks to the network. However, looking up for the reputation information about cloud providers is complex in unstructured P2P networks.

Information lookup. An unstructured topology requires a method to look for the information across the entire P2P network [13]. *Query flooding* is a simple method to look for information by broadcasting queries across the entire network. It is effective in highly-replicated contents but not in rare items. In addition, it has a poor scalability. The addition of probabilistic tables for query routing [12] will increase the efficiency and scalability of unstructured P2P networks.

There are two main reasons to implement our model as an unstructured P2P network instead of a centralised reputation service:

- It is not clear that a subscription fee for such service is suitable for small users, and advertising cannot be targeted to the market participants, which are autonomous computer programs. An open, transparent, and neutral reputation service is economically feasible in P2P networks where the cost of the service is shared among all the participants. The expected benefit by using the service will encourage each participant to bear this cost.

- A centralised reputation service (or a P2P network with central trackers or super-peers) could lead to having central entities that are owned by a dishonest Cloud Provider, which could discard reports that would penalise it or benefit its competitors.

There are currently some centralised Cloud Reputation services that have a clear Business Model (e.g. CloudHarmony [1]), but they are oriented to aggregate all the information of the most popular providers. They provide a set of benchmarks that are executed in such limited set of providers and report their status.

As explained in the introduction, the framework of our research are open Utility Computing Markets [4]. Such markets are dynamic: any small provider can spontaneously enter in the market and they must be automatically considered by the reputation system (for example, a company with few hosts for its internal use may sell its spare resources at night to help amortizing the cost of its infrastructure). This scenario limits the effectiveness of CloudHarmony model because the next reasons:

- The registry of providers must be automatically updated to consider small providers that dynamically enter and leave the market.
- It is expected a large number of providers with a limited number of resources. It is not clear that is technically feasible to execute benchmarks periodically in all the providers. Our model suggests that the previous experiences from other users are actually the benchmarks.
- Services like CloudHarmony consider a limited number of providers that are proven to be honest. In an open Utility Computing market, a dishonest provider could provide a good QoS to the centralized reputation systems and a low QoS to other clients.

3.1 Trust towards a provider

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 communicates 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 [7].

When a client wants to buy a resource, it sends an offer to the providers to start a negotiation and to agree an SLA, which is described as $\{\overrightarrow{S}, \Delta t, Rev(vt)\}$. $\overrightarrow{S} = (s_1, \ldots, s_k)$ 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. $Rev(vt)$ is a revenue function that describes how much money the provider earns or loses after finishing a task. The Violation Time (vt) is the amount of time in which the provider has not provided the agreed QoS to the client. Let MR be the Maximum Revenue, MP be the Maximum Penalty (a negative revenue), MPT be the Maximum Penalty Threshold, and MRT be the Maximum Revenue Threshold, we describe our revenue function as follows:

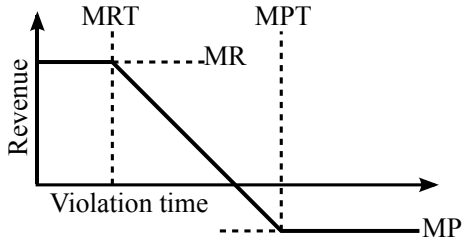

Figure 1: Revenue of a SLA as a function of the violation time (Equation 1).

$$Rev(vt) = \frac{MP - MR}{MPT - MRT}(vt - MRT) + MR \tag{1}$$

Figure 1 graphically represents the possible values of Equation 1. Equation 1 allows a grace period where the provider can violate the SLA without being penalized. If $vt \leq MRT$, the provider will get all the negotiated revenue (MR); if $vt \geq MPT$, the provider will pay the maximum penalty (MP); for $MRT > vt > MPT$ the money to earn or the penalty to pay will be between MR and MP, as a function of vt (see Figure 1). The Maximum Penalty MP is defined to avoid infinite penalties. Client and provider can negotiate the values of MRT, MR, MPT, MP for establishing different QoS ranges that would report different revenues and penalties. Please refer to Section 5 for more details about $Rev(vt)$ and its concrete values. Note that our trust model is independent of this revenue model, which is used for demonstration purposes. This means that the trust model could be used in scenarios with other revenue models, while still being useful for clients and providers.

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 (P_A) about their experiences with B (see Section 3.2, Equation 3). 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 weigh how much importance the client assigns 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 on 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 no economic incentives for the clients. Previous work from the authors [17, 15, 19] show 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 [16]. If a provider is able to guarantee the QoS requirements of a client at lower prices, the client will have incentive to allocate its workloads there; even if the provider has low reputation in factors that are not important to the client.

Considering the above, each client c_x has its own Trust Weight Vector $\overrightarrow{I}(c_x)$, which weighs 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 the provider cp_y is 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 article score the providers according to Equation 2. 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 Weight Vector (element-wise division), and the negative of the magnitude of the resulting vector gives a scoring that shows how trustworthy a provider is for the preferences of c_x. 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.

$$Score(SLA, c_x, cp_y) = -\frac{\left\| \overrightarrow{T}(c_x, cp_y) \oslash \overrightarrow{I}(c_x) \right\|}{Price} \tag{2}$$

The scoring function in Equation 2 will incentivise providers to keep its maximum trust level and, if not possible, to lower prices.

3.2 Defence against reputation attacks

A Cloud Provider can violate an SLA because of technical failures [25], errors in the calculation of the number of resources to provide [15, 19], or dishonest behaviour. The reputation model described in this article 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 reputation attacks towards the system to artificially increase their own reputation and/or decrease the reputation of their competitors. Because our reputation model is decentralised and unmanaged, the clients need a model to prevent reputation attacks 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 with cp_z), the Reputation Trust from c_x to cp_z is calculated as:

$$\overrightarrow{R}(c_x, cp_z) = \sum_{y=1}^{S} \left(T(c_x, p_y^z) \cdot \overrightarrow{D}(p_y^z, cp_z) \right) \oslash \sum_{y=1}^{S} \overrightarrow{T}(c_x, p_y^z) \tag{3}$$

Equation 3 is calculated by asking the peers that have some direct relation with cp_z and weighing 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 based on 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 [28, 11]. Our contribution is complimentary to them, since we deal with the minimisation 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 reputation value which is far from the other reports for the same provider. Note that we tag this as a *potential* bad reputation because, by any reason, an honest peer could have been provided with bad QoS while the others could 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 peer that begins to report different than the other peers. Only repetitive reports that are different will decrease considerably the reputation of a peer.

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 the client is with the specific 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 article we have used a piecewise-defined function that multiplies $T(c_x, p_y)$ depending on how far the trust report was 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 $|a_n - d_n| = \alpha \cdot \sigma_n^2$. If $|a_n - d_n| > \alpha \cdot \sigma_n^2$, the trust relation is multiplied to a minimum of $MAX\_PENALTY < 1$.

Despite of the simplicity of this function to increase or decrease the trust on a peer, it is proven as effective in the evaluation (Section 5). Figure 2 shows that the slope of the linear function that penalises the trust is less pronounced than the slope of the linear function that rewards the trust. In addition, $\frac{MAX\_PENALTY + MAX\_REWARD}{2} < 1$. There are two reasons: (1) the imbalance between $MAX\_PENALTY$ and MAX_REWARD will complicate 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 penalised with severity. Early experiments demonstrated that not defining a function in intervals with different slopes would entail too much instability in the trust updating, and honest peers would lose their trust without good reasons.

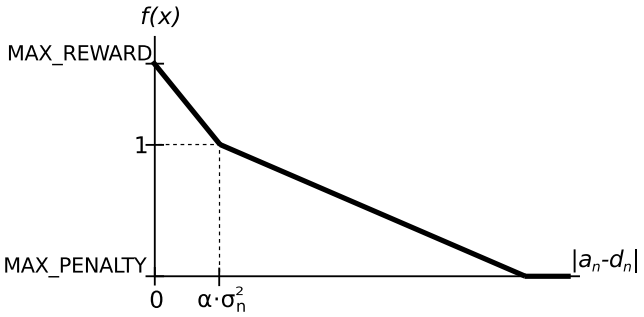

Figure 2: Function to multiply the trust to a given peer, based on its previous report.

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.

4 Reputation-aware SLA negotiation and enforcement

As proven in previous work [16], low reputation decreases the revenue of a provider: the lower trust the less clients will 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 that prices may be dynamically decided in function of many facets, such as number of resources, QoS, client relationship, and market status [15, 17, 19, 18, 20]. This article also defends the need to consider reputation as an additional facet when the provider negotiates an SLA with the client, because adjusting the revenue to the reputation will allow providers to maximise their benefit when reputation is high and sell its resources when reputation is low; in addition, selling the resources when reputation is low allows a provider to recover its reputation.

Pricing in function of 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 several factors: the direct trust, the reputation as reported by all peers, the Trust Weight Vector, and the weights that a particular client assigns to both direct trust and reputation. Direct Trust and Reputation can be approximated statistically, but all the weights are completely private parameters that depend on the preferences of the client.

- *Defining a pricing function.* Each provider must determine the influence of trust in the offered prices. There is lack of information about the prefer-

ences of the clients towards the trust of the providers. Our previous work demonstrated that Genetic Algorithms [18] are suitable to offer prices in markets with hidden information, because they rapidly adapt the pricing function to a changing/unknown environment. However, for simplification purposes, the simulations in this article use linear correlations between reputation and price [16].

A provider does not know the model that each client is using to evaluate and report the QoS, so it is difficult to know how its actions will affect its reputation. But it can know that the higher QoS is provided to a client the higher trust values will be reported to the reputation system; unless the client is behaving dishonestly and reporting false values.

The reputation system in this article is open and peer-to-peer oriented, and allows each of the participants of the system to know the reputation of the provider, but also helps identifying those clients that are reporting false valuations of the service.

In this work, we propose to maximise the reputation as a key objective that will help providers increase their revenue due to the enforcement of the trust relation with their clients. Considering all the actions and policies that a provider can trigger to maximise its reputation, we classify them in two groups:

1. Policies that minimise the impact on reputation derived from SLA violations. This group of policies selectively violates SLAs (to not provide the agreed QoS during a period of time, but restoring it when possible) or cancels SLAs (to not provide any service or resource until the SLA expires).

2. Policies that allocate/redistribute VMs to increase the success rate of the policies of the first group. If a physical node hosts VMs from users with similar trustworthiness, and this node is overloaded, it would be difficult to select which violations have less impact on the reputation, because all the violated SLAs would have a similar impact on the future reputation of the provider. The policies in this article are applied by discriminating/classifying users during provisioning time or redistributing VMs at runtime by migrating them, in order to unbalance the trustworthiness of the users in the same host, and increasing the probability of violating first the SLAs from users with low trustworthiness in case an SLA must be violated.

We consider the policies from the first group: selective SLA violation and/or cancellation: to prioritise trustworthy users under certain extreme situations in which a set of SLAs that are already allocated must be violated temporarily or directly cancelled by some reason (e.g. errors in the resource provisioning process [19, 20] or a hardware failure that makes part of the physical resources unavailable). Such policies would not increase the reputation of a provider, but they would minimise the negative impact of the violations under some unavoidable scenarios (e.g. a system outage).

When the monitoring system of a Cloud provider detects that there are not enough resources to fulfil the QoS of all the VMs in a given node, the process described in Algorithm 2 is triggered.

Data: list of running VMs in a given node
1 order VMs according to a given criterion;
2 n:=0;
3 **while** *workload > 100%* **do**
4 | pause VM n from the ordered list during t seconds;
5 | n:=n+1;
6 **end**

Algorithm 2: Generic algorithm for Selective SLA Violation/Cancellation in each node.

For reputation maximisation policies, the criterion to order VMs (line 1) is the trustworthiness to the client that owns it. To calculate the trustworthiness to a client, the provider can enter in the reputation system as a normal peer, and poll several clients about several providers. If a given client is usually reporting values that are far away from the average, it will be considered unreliable.

The value of t will determine if the SLA is violated ($t <$ time until SLA expires) or cancelled ($t \geq$ time until SLA expires). During normal operation, t is always the time until monitoring data is updated with new values. During other severe problems, such as hardware failures, the VMs running in the nodes with problems are directly cancelled.

In addition to the discrimination of users according to their trustfulness for maximising the reputation of the provider, Algorithm 2 is generic enough to be triggered for achieving other Business-Level Objectives, such as maximisation of the profit or classification of clients according other business criteria [20]. The following section will evaluate the effectiveness of the reputation maximisation policy when compared with the Selective violation of SLAs according to other Business-Level Objectives.

As we will show in Section 5.2.3 the criteria for ordering VMs may be dynamically switched depending on the context of the resources operation. In the evaluation in Section 5.2.3, the provider uses revenue maximisation during normal operation, and switches to reputation maximisation when it detects a hardware failure in the nodes.

We must emphasise that our policy proposes to cancel SLAs only when the provider is not able to fulfil them all. This should be infrequent, used only when the violation is unavoidable, because the economic penalty is paid whatever the client trustworthiness is. For example, when a system outage occurs and a critical part of the hardware resources are unavailable during a period of time, SLA violations cannot be avoided. The idea is at least to minimise the impact on the reputation of the provider. A bad usage of this policy could make the clients lose the confidence in the provider, thus losing profit.

Finally, one might think that a client with a cancelled SLA would be unsatisfied and move to other providers instead of making repeated transactions

in the same provider. This specific study assumes that the clients of the Cloud Market are represented by autonomous, agent-based, decision makers that do not implement the usual human rationality but a mathematical model that calculates whether it is beneficial to repeat a transaction in a provider that did not fulfil an SLA in the past.

5 Experiments

5.1 Simulation environment

The experiments in this article have been performed in simulated environments instead of real testbeds because each single experiment on a real testbed would require to completely occupy a large data centre during days or weeks to get data that is statistically relevant enough. In addition, it would be difficult to get actual users involved in the system to use these resources and provide actual trust reports. We have created a simulator [24] that is available online to facilitate replicating the experiments. Its design comprehends two levels: market and resource fabrics.

Market-level simulation. A marketplace is a service that allows both clients and providers to meet and negotiate SLAs. Our simulated environment does not consider the particularities of the market implementation, and represents the market as a directory of providers and customers where, each time a client wants to buy cloud resources, the following process is triggered: 1) The client sends a SLA template like the one described in Section 3, but excluding the revenue information. 2) The market forwards the client request to the providers. 3) Each provider checks whether it can fulfil the SLA and, in case it can, it calculates the price and allocation that maximises its utility (as discussed in Section 4) and returns the offer to the client. 4) The client asks its peers for the reputation of the providers that returned a price. 5) The client scores all the providers according to Equation 2. It reaches an agreement with the provider whose score is the highest. 6) The client updates its trust to its peers according to Equation 3. When the task is executed, it also updated its direct trust relation to the provider in function of the actual QoS.

Resource fabrics simulation. During the simulation of the allocation and operation of SLAs in a Cloud Provider, the following components operate in parallel:

- *SLA Enforcer* continuously watches the fulfilment of SLAs. If an SLA is not being fulfilled during a time slot, *SLA Enforcer* updates its violation time (Equation 1), which will be considered during the accounting process.
- *Accounting* updates the financial status of the provider when it finishes the execution of an SLA. It stores some statistical information such as profit or losses, number of violations, number of operated SLAs, etc.

- *Virtual Machine,* which takes a portion of the resources (memory, CPU, disk, network ...) of a physical host. At every moment, the simulator accounts the load of its assigned resources. A physical machine can handle a given maximum workload. When the VM has not enough resources to handle the workload (e.g. too much requests during the operation of a Web workload), *SLA Enforcer* is notified.

- *Physical node.* A Cloud Provider owns many physical nodes. Each physical node handles none or several Virtual Machines. When the sum of loads from all the VMs is higher than the total workload the node can afford, some of the VMs will violate their respective SLAs. The number and distribution of such violations would depend on the policies the provider specifies to deal with SLA violations. In addition, a physical node could temporarily fail due to hardware failures. When a physical node fails, all the VMs that were running on it are stopped and, as a result, *SLA Enforcer* handles the violation of their respective SLAs.

The clients look for resources to allocate their tasks in the providers that fit their QoS requirements. The experiments consider three SLOs: CPU, disk, and network bandwidth, which are terms of the Trust Vector and the Trust Weigth Vector.

The simulation can be configured according to some parameters, namely i) the length of the simulated time, ii) the probabilistic distribution of the workloads of the clients (how many clients are accessing the market, how many resources they require at each moment, wether the demand pattern is constant or daily/weekly-variable, and how the resources are linked), iii) the business configuration of the clients (their required QoS, their trust, and their reputation), iv) the parameters of the policies (the thresholds of Equation 1, how the clients weigh the importance of the different types of resources for their applications, and how clients and providers weigh their trust relations between them according to Equation 3), and v) the probabilistic distribution of the failures of resources.

The trustworthiness of the clients follows a folded normal distribution with mean=0.5 and standard deviation=0.2. This means that most clients have trust values near 1 and a few clients are reporting dishonestly. The values for $Rev(vt)$ are: $MRT = 0.05$ (that means that if the agreed QoS is not provided during the 5% of the time or less, the SLA is not considered as violated); $MPT = 0.3$ (when the agreed QoS is not provided during the 30% of time or more, the SLA is completely violated); MR is dynamically established according to $\overrightarrow{S}$ and the market status [15]; $MP = -1.5MR$ (if the provider completely violates an SLA, it must pay back the 150% of the price that the client paid initially). The honest providers normally provide 100% of the agreed QoS during off-peak hours and around 97% during peak hours. The workload follows a web pattern (Figure 3) that varies depending on the hour of the day and the day of the week: there are more requests from Monday to Friday evening than during the late night or the weekend. The Web workload pattern corresponds to the access log

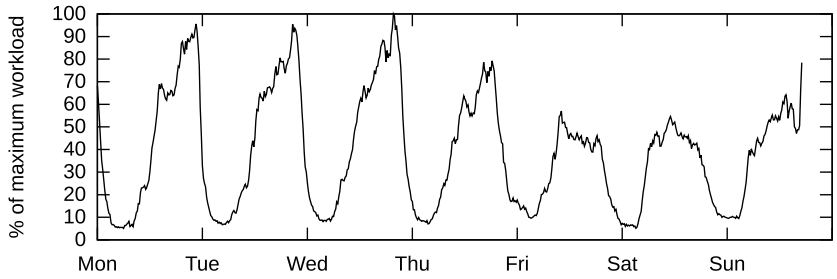

Figure 3: Web workload pattern

of an ISP within our university domain and the exact data source cannot be disclosed due to confidentiality reasons. At the beginning of the experiments, all providers and clients have an initial direct trust of 0.5. The start-up time since the initial trust values converge to their real values is not considered for clients, and it is initially mitigated at provider side with price discounts as a function of the initial trust. Other parameter values are described in their respective experiments.

The simulations rely on constant values that do not intend to reflect real market data, but to evaluate the model in terms of relative values and tendencies. There are no available traces from real commercial Cloud markets that would allow us to realistically adjust the aforementioned parameters. We do not expect to reproduce realistic scenarios, because Cloud markets are yet experimental and we cannot define what is realistic. In the experiments of this article, the values are set according to a main criterion: to provide results that are statistically relevant and proportional to the variations of the inputs. Before the definitive parameters are set, the simulations have been repeated many times to tune some parameters. For example, the maximum load value for the Web workloads must be set to force providers violating enough SLAs to get comparative information between policies. In this case, it is not important to know the exact number of SLAs that a provider violates when it applies a given policy (because it could vary in different market scenarios), but to check whether applying a given policy will decrease the number of violations, and the magnitude of the reduction (e.g. 1-10%, 10-50%, ...).

5.2 Reputation-aware SLA negotiation and enforcement

5.2.1 Effect of hardware failures in reputation

In the first experiment, five providers are competing in a market during 100 simulation steps. As for the remaining experiments whose X axis is measured as *simulation steps*, each step does not represent any time unit, but a cycle that comprehends negotiation in the market, operation and enforcement of SLAs, to see how the operation of SLAs affects in future negotiations. In this group of experiments, the workload is constant at 100% of the resources capacity and

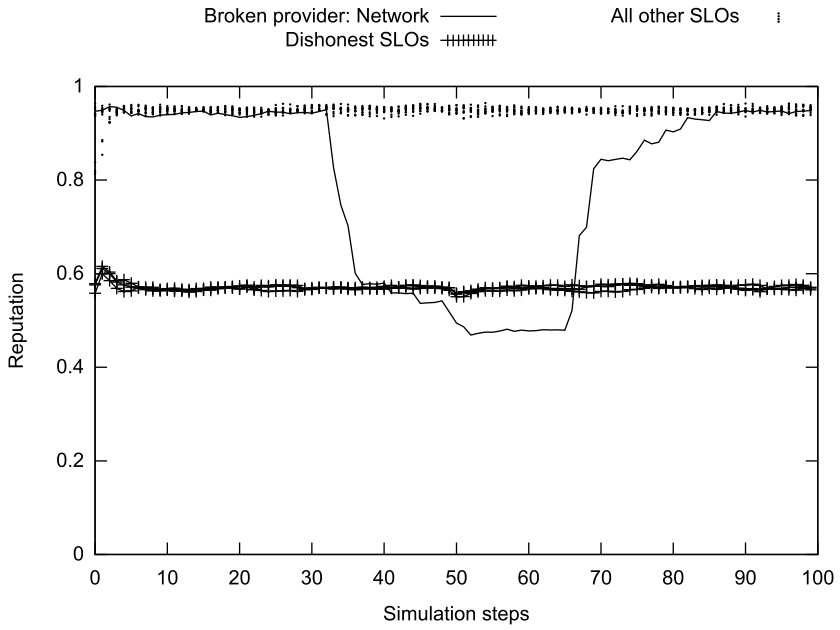

Figure 4: Reputation of providers upon failures and dishonest behaviours.

does not follow the web pattern, since the experiments are focused on measuring the behaviour of the trust model and not the behaviour of the hosts with respect to a given workload.

Four providers are honest and a single 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 [25] in its network at step 33. Therefore, it is providing 50% of its network capacity until step 67.

Figure 4 shows the average trust from the clients to the providers. All the elements of the trust vectors are shown separately, but grouped as follows: 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 4 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.

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	$(0.6, 0.3, 1)$

Table 1: Values of the Trust Weight Vector for each group of clients

5.2.2 Effectiveness of Scoring function during SLA negotiation

To evaluate the effectiveness of Equation 2 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 provider has the maximum reputation in all the SLOs but CPU; the third provider has the maximum reputation in all the SLOs but disk; and the fourth provider has the maximum reputation in all the SLOs but network. A group of 32 clients wants to submit their workloads to the providers, so they score them in function of the trust, the Weight Vector, and the price they ask. The 32 clients are divided in four groups depending on how they weigh 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 2 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 tasks allocation results as follows:

- 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.
- $\sim$50% of tasks from Group 2 are placed in the provider with low network reputation and $\sim$50% in the 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 the provider with low disk reputation, because disk is the SLOs with the lowest importance.

The measured results tend to rounded numbers (e.g. 100% of tasks are allocated in the same provider when the system becomes stable) because the

experiment is repeated in a controlled simulation environment. A real market would add some statistical variability to the results.

5.2.3 Reputation-aware SLA enforcement

To evaluate our reputation-aware SLA enforcement, we have simulated 5 days of a market operation with three types of providers, according to their policy for discriminating SLAs when the resources are overloaded (see Algorithm 2):

- A provider that randomly discards SLAs (*Random*). It is used as a baseline for evaluating how the system would behave without any resource operation policy.
- A provider that discards the SLAs that report less revenue (*Revenue Maximisation*). That has been demonstrated to be effective for maximising the revenue of a provider [15].
- A provider that discards the SLAs from the clients to which there is low trust (*Reputation Maximisation*).

To evaluate the different scenarios, we introduced a global outage of the data centre at day 3 of the simulation. During the outage, the providers only have 20% of their actual resources. The outage has been programmed to happen during a peak workload. That means that about 80% of the allocated SLAs are to be violated during the outage.

Figure 5 shows the evolution of the reputation for the three providers. During off-peak hours, when workload is low, the reputation is near the maximum value for all the providers. During the peaks, the graph shows the effect of the increase of SLA violations in the reputation. While the reputation maximisation policy keeps reputation near 1 during both peaks and off-peaks, the random policy used as baseline clearly shows how the reputation decreases when no policies are applied. Although the revenue maximisation policy does not consider reputation, it indirectly keeps it in the range between random and reputation maximisation policies: the provider tries to pause first the VMs whose SLA violation time is over the Maximum Revenue Threshold (MRT in Equation 1). Therefore, it violates less SLAs and, indirectly, the reputation of the provider is higher than the reputation of a provider that does not apply any policy.

The reputation is much lower in the three providers during the outage around time step 2000 because around the 80% of the SLAs are violated. The difference of reputation between revenue and reputation maximisation is proportionally higher during the outage because the revenue maximisation provider cannot keep SLA violations under MRT.

Figure 6 shows the evolution of the spot revenue over time for the three providers. The revenue is expressed in a fictitious currency, because the revenue information is only important in terms of tendencies and proportions. The figure shows that during normal operation of resources, the revenue of the reputation maximisation provider is slightly lower than the revenue of the revenue

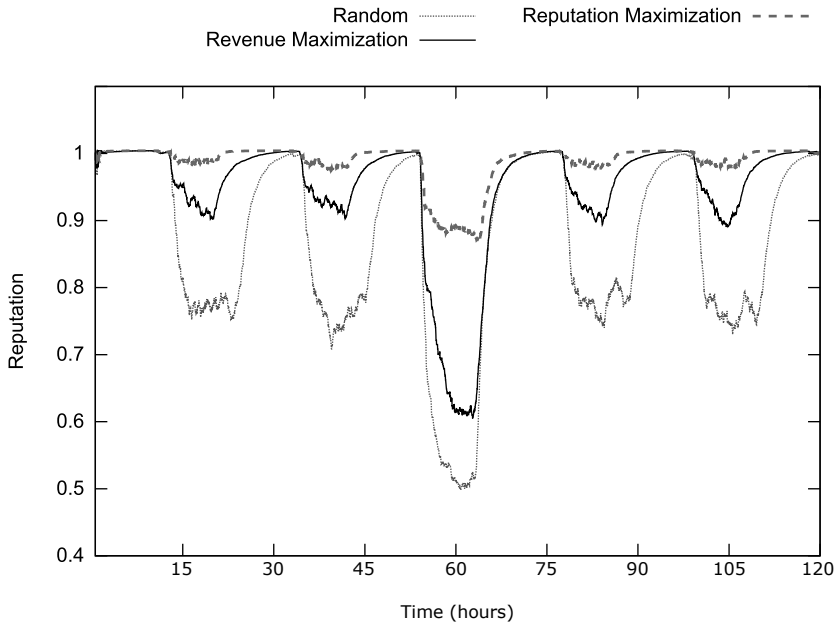

Figure 5: Evolution of reputation for three types of providers.

maximisation provider. A discussion topic comes up from this observation: is reputation maximisation a true Business-Level Objective if it cannot maximise the revenue? The answer could depend on the actual objectives of any organisation. For example, organisations that want to keep service reliability as their differentiating value need to maintain always the maximum reputation over the long term.

However, Figure 6 shows that reputation maximisation policy has a real impact on the revenue of the provider during the outage. After all, the reputation of the revenue maximisation provider is also high during normal operation (> 0.9) and only the outage has a true impact on its reputation.

Considering the previous observations about Figure 6, we have introduced a new provider that is context aware of the environment (normal operation or outage) and dynamically switches the revenue/reputation maximisation policy depending on what is expected to report the highest economic profit. Summarising, the revenue maximisation policy is used when the provider is operating normally; if the monitoring information shows that there is an outage, the provider switches to the reputation maximisation policy and returns back to revenue maximisation policy when the systems are again in normal operation.

The simulation has been repeated with the three providers of the previous section, plus the context-aware provider. In the figures of this section, the random baseline provider has been removed to improve the readability of the figures.

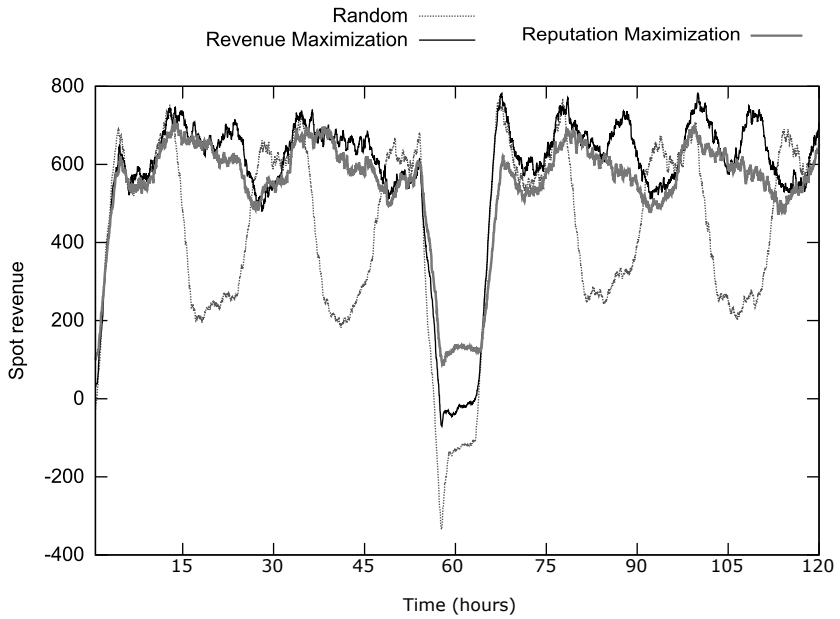

Figure 6: Spot revenue of three types of providers.

Figure 7 shows only a time window of the global simulation, making it easier to visualise. It shows that the context-aware provider maintains a reputation rate similar to the revenue maximisation provider during normal operation (which is high enough to get high revenue) and a rate similar to the reputation maximisation provider during the outage (to minimise the impact of low reputation in revenue).

Our experiments show that the revenue of the context-aware policy is similar to the revenue maximisation policy during normal operation. Figure 8 shows the time window from the outage of the system to the time after the normal operation of the provider has been re-established. Figure 8 shows that the revenue of the context-aware policy is similar to the reputation maximisation policy during an outage and the time after it, while the reputation of the provider is being recovered.

Figures 7 and 8 show that the results for the context-aware policy look slightly different than the corresponding policies because the behaviour of the market varies in different simulations. The market is a complex system and its behaviour is determined by some parameters that are defined using statistical distributions. In addition, it is highly influenced by small differences in the initial status and the actions of the other providers. Our article does not focus on the absolute values of the results, but in the analysis of the tendencies.

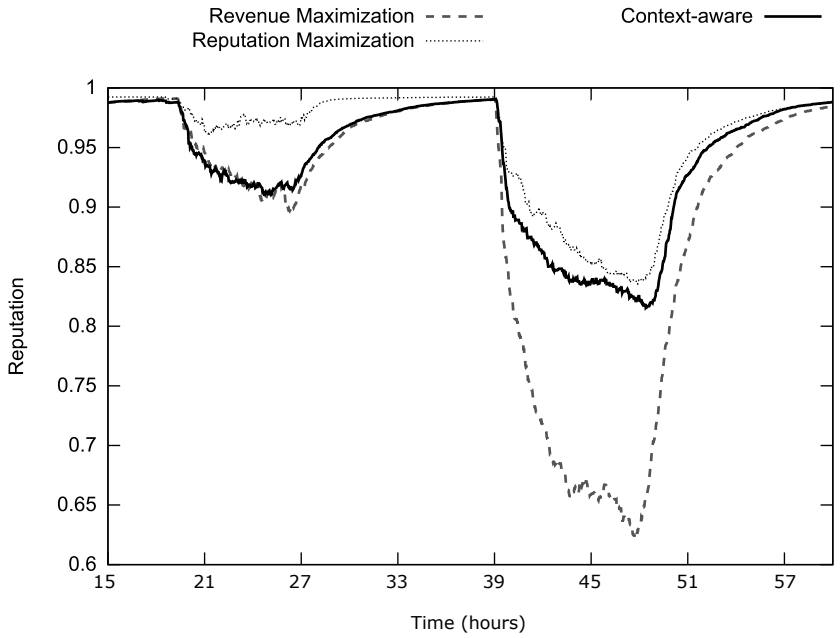

Figure 7: Evolution of reputation for three types of providers, including context-aware policy switching.

5.3 Defence against reputation attacks

This article evaluates and discusses the behaviour of the trust and reputation system towards a given set of reputation attacks [8]:

Self-promoting. A provider that is providing a low QoS infiltrates peers that report high trust values for such provider. An infiltrated peer is a peer that does not have real interest on acquiring Cloud services, but skewing the true reputation of a provider, or a set of providers.

Slandering. A provider infiltrates peers that report low trust values for the competitors, even if they are fulfilling all the SLAs.

Orchestrated attack. A group of peers is coordinated to perform self-promoting, slandering, or both. We will consider a subtype of orchestrated attack called *oscillation attack*: the attackers are divided in two teams. One team behaves correctly to build their confidence from the other peers and the other performs reputation attacks. After a moment, the roles are exchanged to recover the confidence for one group and maximize the impact of the other.

Whitewashing. Dishonest peers abuse the system (self-promoting or slandering) and then use any vulnerability of the system to avoid any decrease

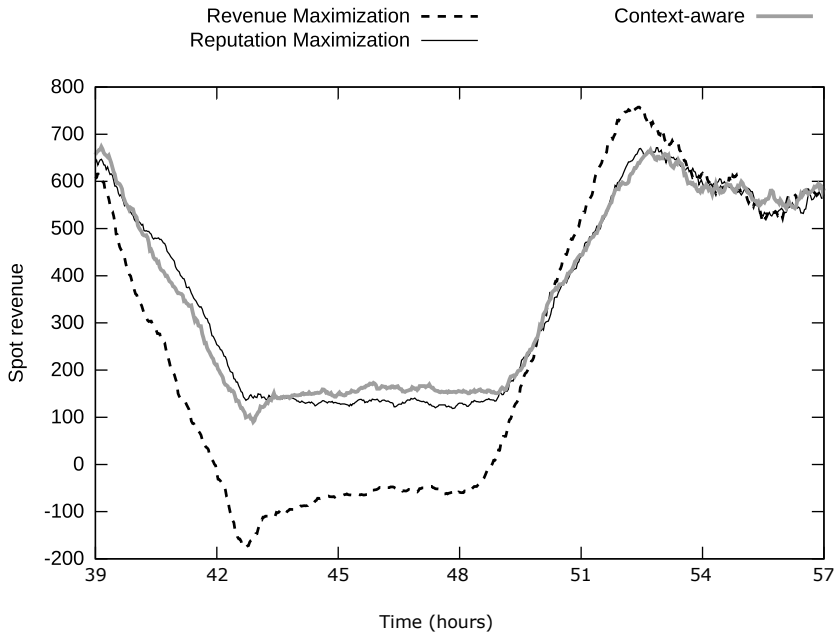

Figure 8: Spot revenue for three types of providers, including context-aware policy switching, during and after an outage.

in their reputation. For example, they leave the system and enter again with a new identity.

Denial of service. After a poor QoS provision, dishonest providers try to prevent the calculation and dissemination of reputation values.

This section experimentally evaluates the behaviour of the system towards self-promoting, slandering, and orchestration attacks. Section 6 discusses the effectiveness of the system towards whitewashing and denial of service attacks.

5.3.1 Self-promoting

In this experiment, the market demand is made of 24 clients that negotiate with the providers for allocating the workloads in the Cloud resources. Before starting a negotiation with a provider, a client asks its peers for the reputation of the provider, then weighs it with its direct trust (if any) and multiplies it by the Trust Weight 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 weighed 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 each peer in function of what they report. This article does not intend to set

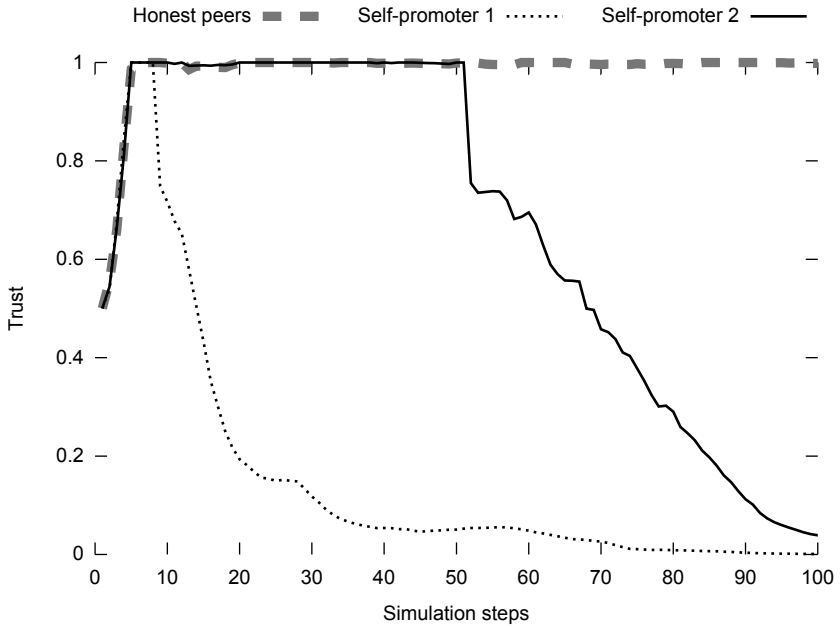

Figure 9: Evaluation of trust to peers that perform Self-promoting.

the optimum values for MAX_REWARD and $MAX\_PENALTY$ constants (Figure 2), 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 should be near 0.5). A peer is performing self-promotion from the beginning of the simulation. The other peer waits until its trust to the other peers is near 1 to maximize the impact of its false reports.

Figure 9 shows that, as initial state, all the peers of a given client have a trust of 0.5. The dishonest client that reports false trust values from the beginning (labelled as Self-promoter 1) increases its trust in the first simulation steps because at this early stage there is no enough statistical information about the self-promoting provider to discard the dishonest peer. After a few simulation steps, the Self-promoter 1 is quickly expelled from the system. In the same way, the Self-promoter 2 quickly decreases its trust as soon as it starts to report false values (step 50).

5.3.2 Slandering

The configuration parameters of this experiment are the same as in the self-promoting experiment. The only difference is that dishonest clients, instead of reporting high values to its provider, report low values to the competitors (the

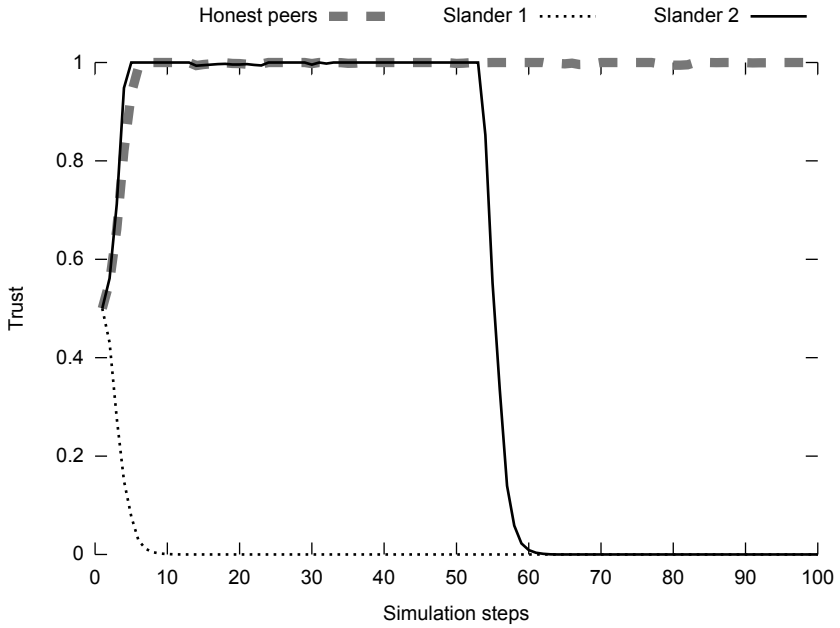

Figure 10: Evaluation of trust to peers that perform Slandering.

50% of their actual QoS).

Figure 10 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, and it is quickly expelled from the list of peers (when it reaches trust 0). The trust to all the other peers is increased, including the second dishonest peer, whose strategy is to increase its reputation to maximise 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 the value 0 at step 59.

5.3.3 Orchestrated attack

In this experiment, a group of dishonest peers is coordinated to perform self-promoting and slandering simultaneously. They report trust values near 1 to the dishonest provider (while it is providing 50% of the agreed QoS) and the 50% of the actual trust to the honest providers.

The dishonest peers are coordinated to perform an *oscillation attack*: the attackers are divided in two teams. During the first 10 simulation steps both groups report true values to increase their trust to the other peers. After this time, the first group starts reporting false values. When the trust of the first group decreases below a given threshold (0.5 in the experiments), they start reporting true values to recover their reputation and the attackers from the second group, who preserved their high trust, start cheating until their trust

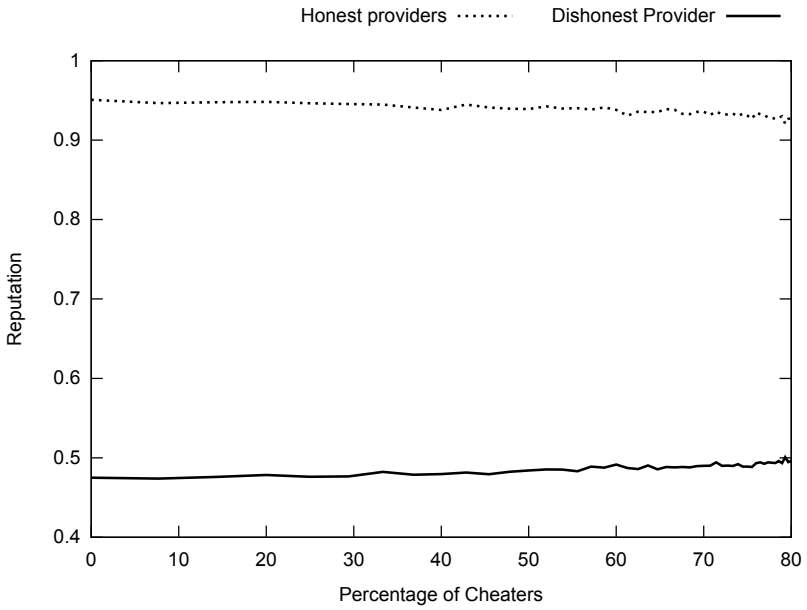

Figure 11: Reputation of providers under oscillation attacks.

falls below 0.5 and the roles are exchanged again. This process is repeated cyclically.

This experiment has been repeated with different honest/dishonest ratio of peers. Figure 11 shows the average trust values for the honest providers as well as the trust value for the dishonest provider. The results show that the influence of the attackers on the real trust values for all the providers is negligible, even when the percentage of cheaters in the system is 80% of the total number of peers (which would be an absurd situation because honest peers would leave the system).

The main reason for the low effectiveness of the attack is that the reputation decreases faster than it is recovered, as previously explained in Section 3.2 (Figure 2). Figure 12 shows the evolution of the trust of the two groups of dishonest peers involved in the *oscillation attack*. Despite their trust oscillates during some time, they finally tend to 0.

Paradoxically, a basic reputation attack without oscillation, where all the attackers are performing the same reports, would be more effective in the unrealistic scenario where the dishonest provider manages to infiltrate a very high number of attackers. Figure 13 shows that that the attackers do not influence the system in scenarios where the honest peers are the majority. When the percentage of cheaters reaches 40% of the total, the system collapses and the reputation values do not reflect the true valuations of the honest peers.

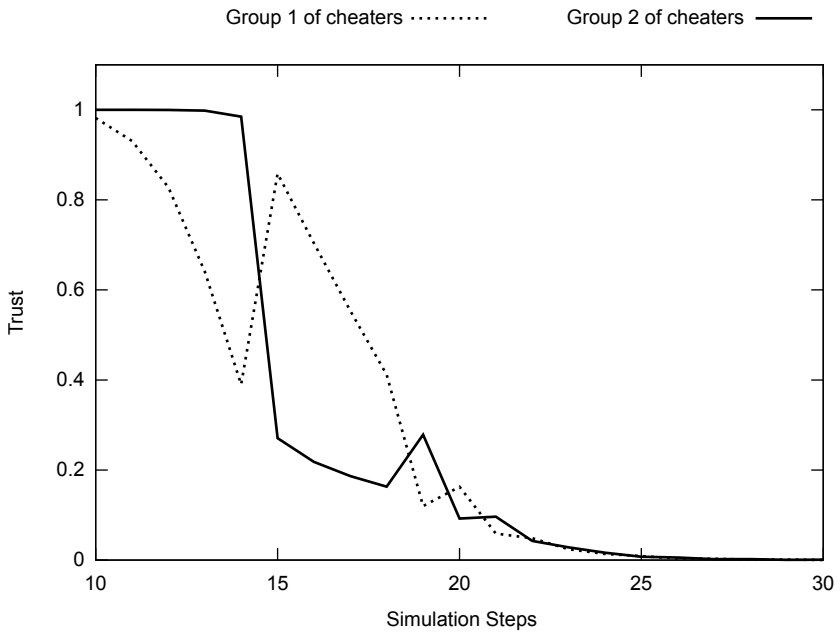

Figure 12: Evolution of trust for peers that perform an oscillation attack.

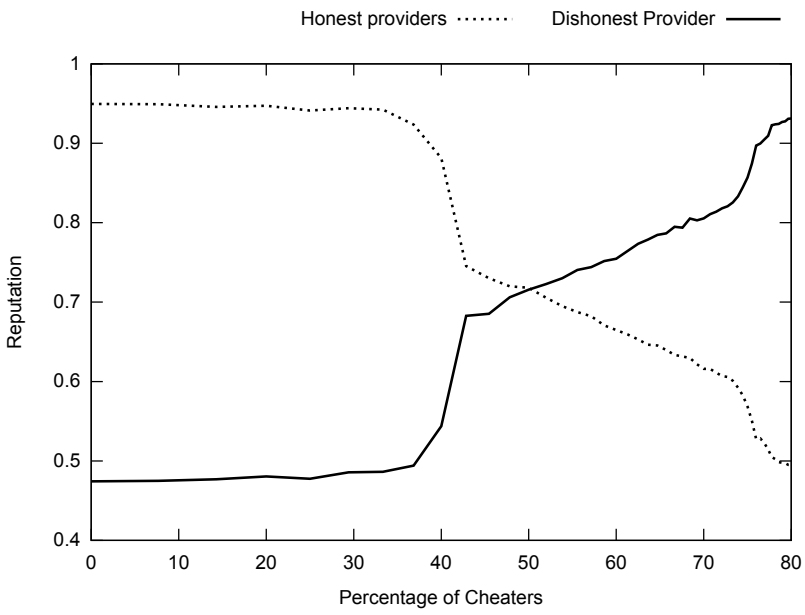

Figure 13: Reputation of providers under a basic orchestrated attack.

6 Discussion: Implementing the model in a real market

This article demonstrates the validity of the reputation model from an experimental point of view. Since this article is focused on the definition of the model, some implementation details are not considered from a formal view. This section discusses the implementability of the model, and the conditions to allow the reputation model being feasible from the trust and economic side.

The first point to discuss is how such a system can exist independent of the service providers or their brokers as a decentralised system. Engaging such a service would require that a consortium of Cloud providers defines and publishes an open protocol for P2P exchange of reputation information and builds a community of users (both clients and providers) that potentially would benefit from using it. This prospective benefit will encourage each actor to assume proportionally the cost to operate the reputation system.

In addition, we identify the following requirements:

- A digitally-signed proof of purchase must be provided by peers that report their trust to a provider. The proof of purchase could be a subset of the agreed SLA, digitally signed by both client and provider. Note that there is no need to store much detailed information about the identity, but simply a unique identifier that remains immutable for every peer. Therefore, a trustworthy Cloud Market requires certification authorities and identity management, which will ensure that outside attackers cannot send trust reports, and will also detect peers that try to change their identity.

- Quantify precisely 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 the 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 [6, 23].

The aforementioned requirements would increase the effectiveness of the system towards reputation attacks. Identity management would minimise the impact of whitewashing if the identities are digitally signed by trusted certification authorities. Attaching the proof of purchase to a report would minimise the coordinated attacks because their economic cost would be high. The decentralised nature of the system would also avoid denial of service attacks, because there is no way that providers prevent the dissemination of the trust values from the peers.

No centralised component bears the cost of implementing our trust model. The cost is shared by all the peers. The cost for each peer, in terms of memory space and extra calculations, is as follows: let s be the number of SLOs in an SLA; let r be the number of peers of a client; let m be the number of Cloud providers. According to the model described in 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 penalise 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 providers, and providers get an economic compensation for possible reputation attacks, as if it were some kind of insurance.

7 Conclusions and Future work

This article describes a reputation model that deals with some open issues in the state of the art. First, we propose a P2P architecture for dealing with the cost of provision of centralised reputation services, which may be a good architecture for other markets but not for Cloud Computing. Second, we define a mathematical model to calculate the trust relation from a client to a provider. This model also defines trust relations between peers and updates them using statistical analysis to detect the trustworthiness of their reports.

In addition to the reputation and SLA negotiation model, we introduce a policy to prioritise users according to their trustworthiness. This policy has a double goal: (1) minimise the impact of the SLA violations in the reputation of the provider and, therefore, in the revenue; and (2) motivate users to report true valuations of the providers. This policy is not intended to motivate a dishonest behaviour but to minimise the impact on the reputation when the violation of an SLA is unavoidable. The common business objectives prevent providers from only establishing SLAs with clients that give them good reviews, or establishing a black list that contains clients with bad reports (even those who are actually honest), because they would lose market share and decrease their profit. In addition, the ignored clients would continue reporting bad reports and the provider would have no chance to restore their trust to them.

The validity of the model is demonstrated through exhaustive experiments. After analysing the policy in comparison with others, we show that providers that behave honestly and apply revenue maximisation policies, in most cases indirectly keep a good enough reputation rate and achieve higher revenue than the providers that apply reputation maximisation. The benefits of reputation maximisation in terms of revenue are noticeable under conditions that imply a high rate of SLA violations. Considering the above, we introduce a new type of provider that switches between reputation or revenue maximisation policies depending on the context. This provider achieves the best revenue in all the cases, and always keeps good-enough reputation rates. Our experiments showed that policies that are unaware of the reputation may have economic losses during system outages while policies that are aware of the reputation keep economic

profits.

This article does not consider the ethical issues of using such policies to cheat the clients with low reputation; also clients that recently joined the reputation network.

A key issue of reputation systems is to motivate their users to report true valuations of the providers. The policies of this article help incentivise such behaviour because clients that report true valuations have high reputation to their peers. Providers that want to keep high reputation will prioritise the QoS for trustworthy clients under certain situations such as peaks of demand or an outage. As a consequence, clients that want to benefit from this positive discrimination will report true valuations of the providers. Since the reputation model is peer-to-peer oriented, any provider could join a network for polling the trustworthiness of a client.

This article also evaluated the behaviour of the system toward reputation attacks. We can conclude that our system behaves reasonably well towards self-promoting, slandering, and coordinated attacks (as experimentally demonstrated in Section 5.3); fulfilling the implementation requirements exposed in Section 6 would avoid whitewashing and denial of service attacks, as discussed in that section.

This article opens a wide range of future work areas: the context-aware provider must be improved with the addition of statistical analysis to dynamically learn how the actions of the provider during negotiation and operation can influence the future reputation. We also plan to improve the policy for selecting the SLAs that are going to be violated. The objective is to achieve a policy that is able to weigh both goals: reputation and revenue maximisation.

Acknowledgements

This work is supported by the Ministry of Science and Technology of Spain and the European Union (FEDER funds) under contract TIN2012-34557, and by the Generalitat de Catalunya under contract 2014-SGR-1051.

References

[1] CloudHarmony. http://www.cloudharmony.com/.

[2] eBay. http://www.ebay.com/.

[3] R. Alnemr, S. Koenig, T. Eymann, and C. Meinel. 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] J. Altmann, C. Courcoubetis, G. D. Stamoulis, M. Dramitinos, T. Rayna, M. Risch, and C. Bannink. GridEcon: A Market Place for Computing

Resources. In *Proceedings of the 5th International Workshop on Grid Economics and Business Models, GECON 2008, Las Palmas de Gran Canaria, Spain, August 26, 2008*, pages 185–196, 2008.

[5] F. Azzedin and M. Maheswaran. Evolving and Managing Trust in Grid Computing Systems. In *IEEE Canadian Conference on Electrical Computer Engineering (CCECE 02)*, volume 3, pages 1424–1429, Winnipeg, Canada, 2002.

[6] I. Goiri, F. Julia, J. O. Fito, M. Macias, and J. Guitart. Resource-level QoS Metric for CPU-based Guarantees in Cloud Providers. In *Proceedings of the 7th International Workshop on the Economics and Business of Grids, Clouds, Systems, and services (GECON 2010), Ischia, Italy*, volume 6296, pages 34–47, August 2010.

[7] M. Gupta, P. Judge, and M. Ammar. A Reputation System for Peer-to-Peer Networks. In *Proceedings of the 13th International workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 144–152, Monterey, CA, USA, 2003. ACM.

[8] K. Hoffman, D. Zage, and C. Nita-Rotaru. A Survey of Attack and Defense Techniques for Reputation Systems. *ACM Computing Surveys*, 42(1):1:1–1:31, Dec. 2009.

[9] R. Jurca and B. Faltings. An Incentive Compatible Reputation Mechanism. In *Proceedings of the IEEE International Conference on Electronic Commerce (CEC 2003), 24-27 June 2003, Newport Beach, CA, USA*, pages 285–292, 2003.

[10] R. Kerr and R. Cohen. Smart Cheaters do Prosper: Defeating Trust and Reputation Systems. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems, Budapest, Hungary*, volume 2 of *AAMAS '09*, pages 993–1000, 2009.

[11] R. Kerr and R. Cohen. Trust as a Tradable Commodity: A Foundation for Safe Electronic Marketplaces. *Computational Intelligence*, 26(2), 2010.

[12] A. Kumar, J. Xu, and E. W. Zegura. Efficient and scalable query routing for unstructured peer-to-peer networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 2, pages 1162–1173. IEEE, 2005.

[13] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, S. Lim, et al. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials*, 7(1-4):72–93, 2005.

[14] D. Ma and A. Seidmann. The Pricing Strategy Analysis for the "Software-as-a-Service" Business Model. In *Proceedings of the 5th international workshop on Grid Economics and Business Models (GECON 2008), Las Palmas de Gran Canaria, Spain*, pages 103–112. Springer-Verlag, 2008.

[15] M. Macias, O. Fito, and J. Guitart. Rule-based SLA Management for Revenue Maximisation in Cloud Computing Markets. In *Proceedings of the 2010 Intl. Conf. of Network and Service Management (CNSM'10)*, pages 354–357, Niagara Falls, Canada, October 2010.

[16] M. Macias and J. Guitart. Influence of Reputation In Revenue of Grid Service Providers. In *Proceedings of the 2nd International Workshop on High Performance Grid Middleware (HiPerGRID'08)*, Bucharest, Romania, Nov. 2008.

[17] M. Macias and J. Guitart. Using Resource-level Information into Nonadditive Negotiation Models for Cloud Market Environments. In *Proceedings of the 12th IEEE/IFIP Network Operations and Management Symposium*, pages 325–332, Osaka, Japan, April 2010.

[18] M. Macias and J. Guitart. A Genetic Model for Pricing in Cloud Computing Markets. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, pages 113–118, TaiChung, Taiwan, 2011.

[19] M. Macias and J. Guitart. Client Classification Policies for SLA Negotiation and Allocation in Shared Cloud Datacenters. In *Proceedings of the 8th International Workshop on the Economics and Business of Grids, Clouds, Systems, and Services (GECON 2011)*, volume 7150, pages 90–104, Paphos, Cyprus, December 2011.

[20] M. Macias and J. Guitart. Client Classification Policies for SLA Enforcement in Shared Cloud Datacenters. In *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 156–163, Ottawa, Canada, May 2012.

[21] N. Miller, P. Resnick, and R. Zeckhauser. Eliciting Honest Feedback in Electronic Markets. Working Paper Series rwp02-039, Harvard University, John F. Kennedy School of Government, Sept. 2002.

[22] O. Rana, M. Warnier, T. Quillinan, and F. Brazier. Monitoring and Reputation Mechanisms for Service Level Agreements. In *Proceedings of the 5th International Workshop on Grid Economics and Business Models, GECON 2008, Las Palmas de Gran Canaria, Spain, August 26,2008*, pages 125–139. Springer, 2008.

[23] G. Reig, J. Alonso, and J. Guitart. Prediction of Job Resource Requirements for Deadline Schedulers to Manage High-Level SLAs on the Cloud. In *Proceedings of the 9th IEEE Intl. Symp. on Network Computing and Applications*, pages 162–167, Cambridge, MA, USA, July 2010.

[24] Reputation-aware cloud market simulator. Online. https://github.com/mariomac/phd/reputation.

[25] R. Tehrani. Amazon EC2 Outage: What the Experts Tell Us. *Customer Interaction Solutions*, 29(12):1, May 2011.

[26] L. Xiong and L. Liu. Peertrust: Supporting Reputation-based Trust for Peer-to-Peer Electronic Communities. *Knowledge and Data Engineering, IEEE Transactions on*, 16(7):843–857, 2004.

[27] B. Yu and M. Singh. A Social Mechanism of Reputation Management in Electronic Communities. In *Proceedings of the 4th International Workshop on Cooperative Information Agents IV-The Future of Information Agents in Cyberspace, Boston, MA, USA, July 7-9*, pages 154–165. Springer, 2000.

[28] J. Zhang. *Promoting Honesty in Electronic Marketplaces: Combining Trust Modeling and Incentive Mechanism Design*. PhD thesis, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, May 2009.